MASTER EN SCIENCES INFORMATIQUES

L'authentification : Etat des lieux et outils d'aide à la décision

Evraud, Geoffrey

Award date:
2016

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2015–2016

L'authentification : état des lieux et outil de décision

Geoffrey Evraud

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Jean-Noël Colin

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Les mots de passe sont souvent à la base des systèmes utilisant l'authentification, ils sont utilisés partout et dans tout contexte. Leur but est de s'assurer que l'utilisateur est autorisé à accéder à une zone protégée d'une application. Ils sont très simples à implémenter et totalement indépendants de toute plateforme ainsi que de toute technologie. Cela en fait un outil particulièrement adopté par la majorité des utilisateurs.

Cependant, ils posent pas mal de problèmes du point de vue de la sécurité. D'autres méthodes existent et possèdent des avantages considérables par rapport aux mots de passe. Mais elles sont difficilement adoptées par les utilisateurs de par leur complexité ou encore par manque de connaissance de leur fonctionnement.
Dans le cadre du mémoire, un inventaire complet des méthodes d'authentification va être établi en pointant du doigt leurs avantages, inconvénients ainsi que leur contexte d'utilisation pour les environnements Web. Ils seront ensuite expliqués en détail.

La suite du document tentera de cataloguer les méthodes d'authentification en distinguant moyens et protocoles pour après s'intéresser en particulier à l'authentification unique.
De ces deux parties découlera un outil d'aide à la décision, qui servira de base lors d'un choix de politique d'authentification au sein d'une entreprise. Celui-ci sera appliqué à deux cas concrets pour en tirer les conclusions quant à sa pertinence et à son utilité.
Un dernier chapitre parlera de la protection des données privées ainsi que de la conscientisation de la sécurité de l'authentification dans les services publics.

Mots clés

Mot de passe, authentification, outil d'aide à la décision, protection des données.

Abstract

Passwords are often the base of systems using authentication. They are used everywhere, in every context. Their goal is to ensure that the user is allowed to access an application protected zone. They are very simple to implement and fully plateform independant, as well as technology independant. That is, they are a tool particulary adopted by the majority of users. However, a lot of problems are reported to them, espacialy in a security point of view. Other methods exist and have significant advantages compared to passwords. But, they are adopted with difficulty by users, either by their complexity, or by a lack of knowledge about the way they are working.

Through this thesis, a whole inventory of authentication methods will be dressed. It will point their advantages, their drawwbacks and their use context in a Web ennvironment. They will be explained in detail afterwards.

The following part of this document will attempt to catalog authentication methods. We will distinguish methods from protocols, and then we will present the Single Sign On approach.
Starting from those two previous parts, we will create an helping tool, which will be used to choice a authentication policy within an entreprise. This tool will be applied to two concrete cases. Some conclusions will be explained compared to its revelance and utility.
A last chapter will talk about privacy and authentication security awareness within public sectors.

Keywords

Password, authentication, helping tool, privacy.

Avant-propos

Ce mémoire représente l'investissement et le travail d'une année entière. Je tiens à remercier toutes les personnes qui ont participé de loin ou de près à la réalisation de ce travail.

Tout particulèrement, je tiens à remercier Monsier Jean-Noël Colin pour m'avoir donné la ligne directrice de ce mémoire ainsi que pour ces précieux conseils. Ils m'ont permis de garder une constante dans mon travail et de visualiser chaque objectif afin d'y parvenir dans les temps.

Je tiens aussi à remercier ma compagne Caroline Annoye pour son soutien durant toute cette année ainsi que mes collègues Xavier Peremans et Ludovic Calmant pour leur relecture et la correction de l'orthographe.

Et enfin, je tiens à remercier mes responsables hiérarchiques, qui m'ont permis de prendre congé selon mon bon vouloir. Ce qui m'a beaucoup aidé pour organiser mes horaires et mon planning.

Table des matières

Glossaire

ID : Identifier ou Identifiant unique

OTP : One-time password

POC : Proof Of Concept

SSO : Single Sign-On

MDSSO : Multi-Domain SSO

FTC : Fail to capture

FTE : Fail to enroll

PKI : Public Key Infrastructure

SP : Service Provider

ASP : Authentication Service Provider

URI : Uniform Resource Identifier

BYOD : Bring Your Own Devices

RP : Relying Party

LDAP : Lightweight Directory Access Protocol

AD : Active Directory

NAP : National Attribute Provider

IDP : IDentity Provider

MW : Middleware

BAP : Business Attribute Provider

CEF : Connecting Europe Facility

DSI : Digital Service Infrastructures

eIDAS : Electronic Identification and Signature

IAM : Identity and Access Management

SPOF : Single point of failure

EID : Electronic Identity Card

OCSP : Online Certificate Status Protocol

CRL : Certificate Revocation List

RRN : National Register

CA : Certificate Authority

IDL : Identity Link

SRA : SourcePin Register Authority

OSTP : Online Secure Transaction Protocol

JSON : JavaScript Object Notation

REST : REpresentational State Transfer

JWT : JSON Web Token

JWS : JSON Web Signature

JWE : JSON Web Encryption

OIDC : OpenID Connect

TGT : Ticket Granting Ticket

ST : Service Ticket

PGT : Proxy Granting Ticket

PT : Proxy Ticket

CAS : Central Authentication Service

WAP : Wireless Application Protocol

ECAS : European Commission Authentication System

STORK : Secure Identity Across boRder linKed

UE : Union Européenne

S-PEPS : SP-country Pan-European Proxy Service

C-PEPS : Citizen Country Pan-European Proxy Service

MS : Member State

V-IDP : Virtual Identity Provider

FAS : Federal Authentication Service

GGA : Gestion des Gestionnaires d'Accès

CHAPITRE 1

Introduction

L'authentification est devenue une phase importante lors du développement d'applications. Celle-ci est souvent réalisée au moyen de simples mots de passe. Cela fait plusieurs décennies que ils sont utilisés comme moyen d'authentification principal. Leur adoption par les informaticiens est souvent dûe au fait qu'ils soient aussi simples à implémenter qu'à déployer. Cependant, l'aspect sécurité n'intervient que très peu dans les cycles de développement ou est complètement ignoré. Ceci pose de plus en plus de problèmes vu le nombre d'applications grandissant sur le Web. De plus, le choix des mots de passe que les utilisateurs prennent n'est pas optimal. Différentes études ont démontré[Bonneau et al., 2015] que malgré le nombre énorme de possibilités de choix, les utilisateurs choisissent souvent un mot de passe très faible d'un point de vue sécurité[Bonneau et al., 2015]. En effet, ces derniers sont souvent un nom, une date de naissance d'un proche ou encore un mot facile à retenir. En outre, leur taille est généralement trop petite. Même si certaines politiques de sécurité supplémentaires peuvent être appliquées, elles ne sont pas toujours suffisantes ou elles sont trop restrictives. Les utilisateurs seront très peu enthousiates vis à vis de celles-ci et tenteront de trouver un moyen de les contourner. Il est très difficile pour les administrateurs de trouver un compromis entre sécurité et utilisabilité pour les utilisateurs. De par leur manque de connaissance de la sécurité ou par un manque de conscientisation, les utilisateurs sont très mal protégés en utilisant un mot de passe.

Il existe une multitude d'attaques possibles visant les mots de passe. Que ce soient par des attaques basiques telles que le phishing, les keyloggers ou des attaques plus évoluées telles que les malwares, voire des rootkits, les hackers ne cessent de tenter d'usurper l'identité des utilisateurs pour en tirer profit. Ces attaques sont un vrai fléau à tout point de vue. Leur impact au niveau économique est énorme pour les sociétés. Tout cette problématique n'est pas une fatalité, il existe heureusement différentes alternatives au renforcement ou au remplacement des mots de passe[Bonneau et al., 2012]. Les scientifiques s'efforcent de créer de nouvelles solutions et de démontrer que le remplacement est possible, parfois même à moindre coût. A défaut de les remplacer, ils proposent également de renforcer l'authentification de façon générale.

Ce travail va tenter de répondre à différentes questions concernant l'authentification, les mots de passe et de leur persistance au sein des systèmes. Comment a évolué la situation les concernant ? La problé-

matique de leur remplacement est-elle toujours d'actualité ? Quelles sont les alternatives possibles ? Les mobile devices sont-ils une opportunité à leur remplacement ? Un SSO global peut-il être envisagé ?
Toutes ces questions permettront de délimiter la recherche de ce travail. Et afin de ne pas partir dans tous les sens, le sujet sera axé autour de l'authentification Web. Nous nous efforcerons de mettre en évidence les possibilités existantes afin de répondre au mieux à ces questions.

Dans la première partie, nous établirons un listing des possibilités d'authentification existantes. Ceci sera réalisé via un tableau de synthèse. Pour chaque méthode, le tableau mettra en évidence les avantages d'un point de vue sécurité, les incovénients, le cas d'utilisation ainsi que l'environnement technique dans lequel la méthode est utilisée. Nous tenterons de différencier le concept derrière certaines méthodes d'authentification telles que les proxies ou le SSO, des moyens d'authentifications eux-mêmes. Cette synthèse de l'existant permettra de poser le cadre afin d'avoir une base de travail pour commencer notre réflexion. Sur base de ce tableau, nous tenterons de différencier les moyens d'authentification comme le mot de passe, le certificat, des protocoles comme le SSO. Nous verrons ce qu'est l'authentification unique et l'authentification forte. A la fin de ce chapitre, nous tenterons d'apporter un élément de réponse aux questions posées dans ce document.
La suite de ce mémoire parlera d'un protocole en particulier : le SSO. Nous analyserons les différents types de SSO existants. Nous verrons qu'il existe deux grandes familles, les True-SSO et les Pseudo-SSO. Chacune d'entre elle s'applique à un contexte particulier, en fonction de différents critères. Nous partirons d'une description simple pour ensuite aller plus en détail avec chacun d'entre eux. Nous prêterons une attention particulière à distinguer protocoles, principes et implémentations utilisant ces mêmes protocoles. Nous réaliserons ensuite une comparaison de ceux-ci afin de déterminer dans quel contexte il serait préférable d'utiliser un type de SSO plutôt qu'un autre.

Dans le chapitre suivant, un outil d'aide à la décision sera créé dans le cadre de l'authentification. Les chapitres précédents serviront de base lors de la création de ce dernier. Il sera présenté sous la forme d'un arbre dont chaque arc sera un critère de choix. Au fur et à mesure des choix effectués, l'utilisateur descendera dedans jusqu'à arriver à une feuille. Celle-ci représente une méthode ou un protocole d'authentification. Le but de cet outil sera de pouvoir le présenter à un responsable informatique d'une entreprise dans le cadre d'un choix de politique d'authentification.
Afin de tester l'utilité de cet outil, il sera appliqué à deux cas concrets. Le premier cas est l'université de Namur dont le responsable informatique souhaiterait améliorer son système de SSO actuel. Le deuxième cas est dans le cadre de notre travail. La personne s'occupant de l'infrastructure IT est intéressée par ce mémoire afin de valider ses choix antérieurs. Une fois présenté, ils pourront tous deux évaluer leur politique d'authentification actuelle et émettre des remarques. Par la suite, des conclusions seront tirées par rapport à la pertinence de l'outil. Les différentes critiques seront prises en compte afin de pouvoir l'améliorer.

Après ceci, nous présenterons la problématique concernant le respect des données privées des utilisateurs. Les descriptions montreront que les différentes méthodes se basent sur les données utilisateurs lors

des flux d'authentification. Nous verrons que leur utilisation ne se fait pas toujours en bon père de famille et que la sécurité qui y est liée n'est pas toujours optimale. Les grandes règles et principes de l'OCDE seront présentées ainsi que les lois de la Commission Européenne. Nous discuterons de la conscentisation des services publiques dans ce cadre et des différentes solutions existantes comme CEF eID et CSAM. Dans la section suivante, nous parlerons de l'évolution de l'authentification sur le Web en émettant une distinction entre PC et mobile devices. Nous verrons que des méthodes émergeantes sur le marché proposent une alternative entre coût, sécurité et utilisabilité. Cela nous permettra d'apporter une réponse par rapport aux questions posées en début de ce document.

Enfin, pour clôturer ce travail, une conclusion reprenant l'ensemble des éléments clés du travail sera réalisée. Elle discutera de la démarche qui a permis d'obtenir des réponses aux questions que nous nous posions, des difficultés rencontrées lors de la réalisation de ce document, ainsi que l'apprentissage que nous avons pu en retirer.

CHAPITRE 2

Etat de l'art

2.1 La problématique des mots de passe

Depuis plusieurs décénies, les mots de passe ont été utilisés comme outil principal permettant de protéger certaines ressources privées des applications. S'il est très facile de les implémenter et de les utiliser, leur sécurité en contre-partie, n'est pas optimale. Même s'ils sont utilisés par la majeure partie des authentifications, les chercheurs s'efforcent de démontrer qu'ils sont peu sécurisés et qu'ils sont sujets à une multitude d'attaques. Outre les attaques existantes, des individus peuvent simplement tenter de les deviner. Celà a été le cas au début de leur implémentation, puisque "certaines personnes s'amusaient à deviner le mot de passe de leurs collègues". Le fait de deviner facilement un mot de passe est dans la majorité des cas dû au fait que ce dernier est trop faible. En effet, il a été mainte fois démontré que les utilisateurs ne choisissent pas un mot de passe aléatoirement. Nous savons que l'espace de choix d'un mot de passe est gigantesque et que malgré cela, la majorité des utilisateurs choisiront quelque chose qui est facilement mémorisable. Un prénom, une date de naissance, le nom d'un animal, etc.... sont des exemples courants de secrets trop simples.

Claude Shannon[Bonneau et al., 2012] a défini un terme pour mesurer la force des mots de passe. Ce dernier s'appelle l'entropie. Sa dénition nous dit "*qu'elle est basée sur une petite distribution uniforme de choix de mots de passe dans un espace de choix existant gigantesque. Le logarythme de la taille de cette distribution représente l'entropie.*" Cette dernière permet d'estimer la force d'un mot de passe. C'est d'ailleurs ce que proposent certains sites en mettant à disposition des utilisateurs, une échelle à degrés ou graduée, qui donne une idée de la force du mot de passe lors du choix de celui-ci pendant une inscription. Cela permet de les influencer vers un choix plus renforcé. Ils ont quand même tendance à prendre en compte cette échelle et à choisir quelque de chose de plus complexe que la normale. Cependant, il est tout à fait compréhensible qu'un utilisateur ne sera pas enclin à choisir un mot de passe comme "dxk156adsQ?J!". Il s'orientera vers plutôt quelque de chose qu'il puisse retenir aisément.

Des solutions existent afin de pallier à la faiblesse de ces mots de passe. Les administrateurs systèmes ont par exemple mis en place des politiques d'authentification renforcées. Ces dernières empêchent sou-

vent de choisir un mot de passe dont la taille est plus petite que huit caractères [1] ou qui ne comprenent pas une majuscule et un chiffre. Mais malgré tout ceci, nous sommes forcés de constater que les utilisateurs tentent de contourner le système, par exemple, en incrémentant le chiffre dans leur mot de passe ou en changeant une lettre.
Dès lors, d'autres méthodes d'authentification que le mot de passe sont apparues au fur et à mesure du temps. Même si elles possèdent des avantages non négligages d'un point de vue de la sécurité, elles ne remplacent toujours pas les mots de passe.

En prenant du recul par rapport à l'historique décrit ci-dessous et par rapport au sujet de l'authentification en général, nous nous sommes posés plusieurs questions :

- Les mots de passe sont-ils toujours aussi présents qu'avant ?
- Quelles sont les alternatives à leur remplacement ?
- Est-il possible de les remplacer totalement ?
- Un déploiement de masse des nouvelles méthodes plus sécurisées est-il envisageable ?
- Quel est l'avenir des mots de passe, de leur présence sur le Web ?
- Quel est l'avenir de l'authentification et de sa sécurité ?
- Avec l'expansion des smartphones et des tablettes, les mobile devices ne sont-ils pas une opportunité de les remplacer définitivement ?
- Existe-t-il réellement une solution viable à ce problème ?

De manière générale, nous nous interrogeons par rapport à la présence accrue des mots de passe lors de la phase d'authentification sachant qu'ils sont peu sécurisés.
La suite de ce document tentera d'apporter une réponse la plus complète possible à ces questions.

2.2 Le Web et son évolution

Nous savons que les mots de passe sont utilisés par beaucoup de systèmes et de protocoles. C'est notamment le cas des protocoles SSH, FTP, SMTP ou HTTP [2]. L'authentification est omniprésente au sein des systèmes. Ce terme est souvent utilisé à toutes les sauces dans la littérature et au travers le Web. Il est d'ailleurs fréquemment confondu avec le terme identification. Or, ces deux termes représentent deux concepts différents mais complémentaires.
Le principe de l'authentification est de prouver qu'on est bien la personne qu'on prétend être. Selon le cours de sécurité de Mr. Colin[Colin, 2016], il existe trois façons de prouver qui nous sommes :

1. Par ce que nous connaissons.
2. Par ce que nous sommes.
3. Par ce que nous possèdons.

1. De façon générale, huit caractères est un choix très courant.
2. Liste non exhaustive.

L'identification est quant à elle, la phase durant laquelle le système associe un ID, un nom, une adresse ou tout autre attribut à la personne présente dans le système.

De nos jours, quasiment toutes les applications implémentent une authentification lorsqu'elles doivent protéger certaines ressources. Elles sont divisisées deux grandes familles : celles qui sont dites 'lourdes' et les autres, dites 'légères'.
La première famille concerne les applications installées directement sur la machine cliente. Elles sont souvent déployées via des packages et nécessitent une configuration particulère sur les machines qui les hébergent.
La seconde famille concerne les applications qui peuvent être accédées à travers un navigateur Web. Elles sont légères par le fait que la machine cliente ne nécessite quasiment aucune configuration afin de pouvoir fonctionner [3]. Il est souvent dit de ces dernières qu'elles sont orientées Web.
Ces dernières années, nous avons vu l'Internet grandir de façon exceptionnelle. De ce fait, les développements orientés Web ont suivi cette tendance. Les applications lourdes deviennent de plus en plus abandonnées pour laisser place à l'orienté Web. Dès leur apparition, l'authentification par mot de passe a été le standard utilisé afin de protéger les zones privées et les données sensibles de ces dernières.

Nous allons axer notre étude vers le cadre Web étant donné que la majorité des développements des applications ont pris cette orientation. Implicitement, cela implique donc aussi que la majorité des attaques visent ces applications.
Le cadre Web représente l'ensemble des applications accessibles en HTTP(S) depuis un navagiteur Web sous une machine ou mobile devices (smartphones ou tablettes). Le terme Web va même plus loin puisque Web en anglais veut dire 'toile' en français. Cela fait penser à une toile d'araignée où tous les fils de la toile sont reliés entre eux. Ce principe est le même pour les applications présentes sur Internet. Le Web implique que tout est connecté d'une manière ou d'une autre. Les applications Web ont donc la possibilité d'échanger des informations entre elles. Cette tendance est d'ailleurs bien présente de nos jours via l'apparition d'applications de type WebService à cet effet.

C'est indéniable, le Web a évolué de manière exponentielle ces dix dernières années. Cependant, nous remarquons une nouvelle tendance qui fait son apparition, celle des mobile devices comme les smartphones et les tablettes. Certaines sites voient une augmentation conséquente de leur traffic Web par ces appareils. Le problème de l'authentification pour ces devices est le même que celui pour les machines classiques. Cependant, les utilisateurs ont tendance à éviter d'entrer des mots de passe via leur smartphone. Ceci est simplement explicable par le fait que ce n'est pas aisé d'entrer un mot de passe avec un clavier de petite taille. Nous sommes donc en position de se poser la question da la viabilité de ces mots de passe pour ces appareils. Si les utilisateurs ne sont pas très enclin à les utiliser avec un clavier très petit, ne pourrait-on les remplacer pour une méthode plus simple et plus sécurisée ? Et comme nous l'avons cité ci-dessus : ne peut-on pas voir ici une opportunité des les remplacer définitivement ?

3. En fait, elles nécessitent souvent des programmes comme JAVA ou Flash.

2.3 Tableau des méthodes d'authentification

Afin de répondre aux questions que nous venons de soulever, nous allons établir un listing de l'ensemble des méthodes d'authentification existantes. Nous avons synthétisé ces méthodes dans le tableau 2.1 présent à la page suivante. Les colonnes du tableau représentent les critères de comparaison, tandis que les lignes représentent les méthodes d'authentification. Nous avons regroupé les avantages tels que le coût, l'utilisabilité et le déploiement en un seul point. Ceci représente le premier critère. Un système qui est trop coûteux, difficile à déployer ou encore trop contraignant pour les utilisateurs sera peu voire pas adopté.

Le second critère est l'avantage d'un point de vue de la sécurité. Autrement dit, le système est-il assez sécurisé pour être utilisé ? Ce dernier est important puisque nous cherchons à remplacer les mots de passe par une méthode d'authentification plus sécurisée.

Si nous avons parlé des avantages, il existe aussi des inconvénients. Aucun système n'est parfait. Nous avons donc assigné un troisième critère à ce tableau représentant quelconque(s) inconvénient(s) liés à la méthode sélectionnée.

Le quatrième critère est le cas et/ou le secteur d'utilisation du système. Certains systèmes peuvent être destinés à la recherche tandis que d'autres sont plus utilisés dans des secteurs commerciaux, civils ou encore dans le cadre familiale et privé.

Et enfin, le dernier critère est l'environnement technique. La solution peut-être utilisée en interne ou en externe. Elle peut aussi être liée à un protocole particulier. Il est intéressant de remarquer que certains systèmes d'authentification embarquent d'autres systèmes d'authentification ou bien leur principe dans leur fonctionnement.

Certains termes sont utilisés dans ce tableau, nous allons en définir la sémantique :

Simple : d'un point de vue utilisabilité, ce terme veut dire que l'utilisateur peut utiliser le système sans difficultés majeures.

Mature : le système est mature à partir du moment où il a été implémenté, testé , corrigé et validé par une communauté importante d'utilisateurs durant un certain nombre d'années.

Peu coûteux, coût négligage : le coût d'implémentation et d'utilisation du système est faible par utilisateur.

Portable : le système est indépendant de l'insfratucture l'hébergeant. Il peut être déplacé très facilement.

Intuitif : l'apprentissage du système se fait de manière très simple.

Déploiement : mise en place du système.

Déploiement interne : mise en place du système dans le but d'être utilisé par les utilisateurs internes.

Déploiement externe : mise en place du système dans le but d'être utilisé par les utilisateurs externes.

Web : le système interagit avec le monde extérieur via requêtes HTTP(S).

Canaux sécurisés : canal utilisant SSL ou TLS tels que SMTPS, FTPS, SSH, etc...

TABLE 2.1 – Ensemble des méthodes d'authentification existantes

Type d'authentification	Avantages coût, utilisabilité et déploiement	Avantages sécurité	Inconvénients	Cas d'utilisation	Environnement technique
Mot de passe Web	Simple Portable Indépendant de la technologie Coût négligeable	Possibilité de chiffrement + sel	Peu sécurisé Faible entropie	Pour tout type d'authentification	Utilisable par tout protocole Déploiement interne et externe
Password Manager	Simple Unique mot de passe à entrer Portable grâce au Cloud Coût négligeable Possibilité de SSO local	Possibilité de chiffrement	Lié au navigateur (plugin) Attaque sur le mot de passe principal (SPOF) Pas de recovery from loss Sensible aux keyloggers , malwares et spywares	Utilisable pour tout type d'utilisateur	Web Déploiement interne
Proxy-based	Déploiement facile Coût négligeable Possibilité de SSO	Login sécurisé Possibilité de ne pas stocker les mots de passe Possibilité de chiffrement Possibilité d'utilisation d'OTP	Pas toujours simple pour l'utilisateur Sensible aux sessions hijacking Mécanisme de révocation de code pour certains proxies Conversation d'une liste de code côté utilisateur pour certains proxies Configuration du navigateur pour les proxies HTTP	Permet de s'authentifier d'une machine publique Utilisé dans le cadre de la recherche Utilisé en tant que 3rd party	Web, FTP, canaux sécurisés Déploiement interne et externe
Biométrie	Mature	Non répudiation Présence physique	Coûteux Déploiement complexe Taux de faux positifs et négatifs Performances pas toujours très bonnes Sujet à des attaques de rejeu d'empreinte	Utilisé dans plusieurs cadres : Commercial Civil Forensic Gouvernemental	Web Déploiement interne
Phone-based	Peu coûteux	Résistant au phishing	Nécessite un mobile device de confiance Configuration navigateur requise Gestion des certificats requise	Permet de s'authentifier d'une machine publique Utilisé dans le cadre de la recherche	Web Déploiement interne
SSO	Simple Mature Cerains SSO ne sont pas liés à la technologie	Authentification unique gérée par un serveur dédié Gestion des comptes et droits simplifiée Certains SSO ne transfèrent pas les mots de passe aux applications	Sensible aux attaques volant le mot de passe principal Peut nécessiter des configurations lourdes côté applicatif Peut être relativement coûteux	Utilisé dans plusieurs cadres : Entreprise Gouvernemental Machine locale d'un utilisateur	Web, canaux sécurisés Déploiement interne et externe
Token Hardware	Simple Mature	Authentification via OTP Résistant aux attaques par rejeu Encryption	Nécessite un support physique en permanence Coûteux Renouvelement du support physique parfois difficile	Utilisé pour des authentifications fortes (banque, armée, ...) Utilisé comme second facteur d'authentification	Web, canaux sécurisés Déploiement interne et externe
Authentification multi-facteurs	Mature	Résistant au vol d'identifiants Permet de chainer les moyens d'authentification	Utilisation parfois éprouvante	Utilisé pour des authentifications fortes (banque, armée,...)	Web, canaux sécurisés Déploiement interne et externe
Authentification graphique	Peu coûteux Simple Intuitif	Résistant aux attaques de rejeu , au guessing et au brute-force Ne divulgue pas le secret	Pas mature Pas utilisable pour les non-voyants Peut-être prédictible dû aux patterns choisis par les utilisateurs Nécessite une configuration serveur Parfois de shoulder-surfing	Utilisé dans la recherche Utilisé dans le secteur commercial	Web Déploiement interne et externe
One-time password	Peu coûteux Simple Mature	Résistant au phishing, malwares et spywares	Nécessite un support porté par l'utilisateur Nécessite une configuration serveur	Utilisé par les banques, les gouvernements, VPN Utilisé par d'autres types d'authentification (RSA, phoned-base, proxy-based,...) Utilisé lors de l'authentification multi-facteurs	Web, canaux sécurisés Déploiement externe
Certificat	Peu coûteux Mature	Résistant au shoulders surfing, au spywares, brute-force, guessing	Nécessite un gestion des certificats côté utilisateur et serveur	Utilisé pour tout type d'authentification	Web, canaux sécurisés Déploiement interne et externe
FIDO	Simple	Authentification local sur le device Résistant à la majorité des attaques classiques Basé sur la cryptographie Protection de la vie privée	Nécessite un support porté par l'utilisateur Nécessite une configuration serveur	Utilisé pour tout contexte Web nécessitant une sécurité élevée	Web Déploiement interne et externe

Ce tableau liste l'ensemble des méthodes d'authentification existantes. Cependant, ceci est une synthèse un peu brute. Or, nous pouvons distinguer le moyen d'authenfication, du protocole d'authentification lui-même. En effet, les moyens d'authentification représentent simplement la méthode utilisée, comme par exemple le mot de passe. Le protocole, quant à lui, se base sur une ou plusieurs méthodes ou sur des concepts bien spécifiques.
En prenant le tableau comme entrée, nous allons décrire si la méthode appartient au 'moyen d'authentification' ou à un 'protocole d'authentification' . Nous allons ensuite mettre en avant les avantages de chacun par rapport au cadre Web que nous avons défini ci-avant.

2.4 Les moyens d'authentification

2.4.1 Le mot de passe

Le mot de passe est la méthode la plus utilisée lors d'un choix d'authentification Web. L'explication en est très simple, il comporte beaucoup d'avantages pour les développeurs des systèmes ainsi que les utilisateurs.
Tout d'abord, son coût est très faible, tant au niveau développement qu'au niveau serveur. Ensuite, son utilisation est intuitive. L'utilisateur n'a besoin d'aucune formation pour pouvoir remplir un formulaire lui demandant ses identifiants. Troisièmenent, son déploiement est simple et permet de mettre en place un système d'authentification très rapidement. De plus, il n'est aucunement lié à une technologie, un langage de programmation ou encore une plateforme. Nous pouvons noter que du côté serveur, le couple identifiant / mot de passe est souvent stocké dans un annuaire et/ou une base de données. Ceci peut donc entrainer des coûts supplémentaires mais cela reste faible.
Le mot de passe présente un inconvénient majeur : la sécurité.
Il est sujet à un ensemble de type d'attaques. Voici les plus connues :

- Les attaques de type brute-force, testant toutes les possibilités existantes. Si par exemple, l'utilisateur est "admin" et le mot de passe "admin123". L'attaque par brute-force trouvera le mot de passe en quelques minutes.
- Les attaques sur base d'un dictionnaire. Ce dernier est une liste des identifiants les plus connus. Chaque entrée de la liste sera testée dans le formulaire du site.
- Les attaques de types phishing, eavesdropping, malwares ou spywares. Le principe de ces attaques est de voler les identifiants soit via un logiciel installé sur la machine cible, soit via vol lors d'une phase d'authentification.
- Les attaques basées sur le social engineering. Dans ce cas-ci, l'attaquant va tenter par diverses moyens de récupérer des informations auprès de l'utilisateur. Il pourra alors deviner ou récupérer le mot de passe de la cible via la question secrète.

- Les attaques basées sur le fait que l'entrée soit mal filtrée. Par exemple une injection SQL :
  Une requête classique :

  ```
  select user , pwd from users where user = "xxx" and pwd ="yyy";
  ```

  Si l'attaquant passe admin"#, la requête devient :

  ```
  select user , pwd from users where user = "admin"#" and pwd ="yyy";
  ```

  L'authentification pour l'utilisateur "admin" sera alors validée, et ce, peu importe le mot de passe, puisque ce dernier sera commenté dans la requête SQL.

La liste de ces attaques ne représente pas toutes les possibilités existantes mais cela démontre une certaine faiblesse des mots de passe. Celle-ci est d'ailleurs accentuée par l'entropie de ceux-ci ainsi que leur utilisation multiple au niveau de plusieurs sites Web.
Dans ces conditions, un attaquant a de grandes chances de compremettre ceux-ci. Un mot de passe d'une complexité plus grande pourrait augmenter l'entropie et donc réduire le nombre de vols d'identifiants. Mais cela comporte néanmoins un effort de mémorisation supplémentaire pour les utilisateurs.
Des mesures de sécurité supplémentaires existent et permettent de contrer certaines attaques telles que l'utilisation protocole SSL/TLS permettant le chiffrement, l'ajout d'un sel, ou encore des politiques de renforcement de mot de passe. Cependant, beaucoup de sites ont été mis en échec malgré ces mesures, et ce, à tout niveau de business.

2.4.2 One-time password

Les OTPs font partie des méthodes d'authentification fortes. Ils sont généralement utilisés en tant que second facteur d'authentification[Haller, 1995] par le biais de devices externes comme les tokens hardwares, les smartphones, listes papiers, etc... Les sites Web tels que ceux des banques [4] sont friands de cette méthode et l'utilisent couramment. Comme le dit leur nom anglais, les OTPs sont des mots de passe temporaires. Leur durée de vie est généralement de 30 ou 60 secondes.
Un flux OTP implique deux acteurs : le client avec son device et le serveur.
Lors de cet échange, le but sera de prouver que le client connait le secret sans pour autant que le faire transiter sur le réseau. Pour arriver à cela, deux phases sont nécessaires :

- La première est la génération des codes OTP's. Durant cette phase, le système va créer la séquence de codes. Elle se compose de deux étapes :
  — Dans la première étape, ou étape d'initialisation, l'utilisateur fournit une entrée ou secret partagé sous forme d'une pass-phrase [5] au système. Celle-ci est concaténée avec une seed [6]. Le résultat de ceci est alors passé dans la fonction de hashage et ensuite réduit à une expression

4. Nous parlons ici du PC banking.
5. Généralement, c'est l'utilisateur qui fournit l'input mais cela est automatisable.
6. Une seed est une suite de caractères permettant d'avoir un point d'accord entre serveur et client (device) afin de créer séquence commune.

de X bits. Pour l'exemple, nous prendrons une chaine de 64 bits mais le nombre de bits peut être différent (32,128,256,...).

— La deuxième est l'étape de calcul. Durant celle-ci, le système va utiliser le résultat créé en première étape comme entrée afin de calculer une séquence d'OTPs. Pour ce faire, la fonction de hashage va être appliquée un certain nombre de fois, produisant N OTPs. Le premier OTP correspondra alors à N fois l'application de la fonction. Le second OTP N-1 fois et ainsi de suite.

- La seconde phase est l'authentification proprement dite. Certaines conditions sont obligatoires pour qu'elle se réalise correctement :
  — Le device utilisé par le client et le serveur doivent utiliser le même algorithme de hashage, sans quoi il sera impossible pour le client et le serveur de se comprendre correctement. Pour utiliser une métaphore, nous pourrions dire qu'ils doivent parler la même langue.
  — Le client et le serveur doivent se mettre d'accord sur une séquence commune pour faire la vérification du code entré. Ils doivent être synchronisés. Ceci est réalisé via un challenge envoyé par le serveur à l'utilisateur. Ce dernier est constitué d'une séquence de nombre et de la seed. Il permet au générateur de code OTP de récupérer les paramètres nécessaires pour calculer le bon code OTP à partir du secret partagé.
  Le challenge se présente sous ce format[N. Haller, 1998] :
  OTP-<ALGORITHM IDENTIFIER> <SEQUENCE INTEGER> <SEED>
  Une fois les paramètres extraits, la seed pourra être utilisée pour synchroniser la séquence et l'utilisateur pourra entrer le code reçu. Pour vérifier que ce dernier est correct, le système dispose d'une base de données. Pour chaque utilisateur, elle contient le dernier code correspondant à une authentification réussie, ainsi que le premier code OTP de la séquence initiale. Lors de l'authentification, le serveur va décoder le code OTP entré par l'utilisateur en 64 bits et ensuite lancer la fonction de hashage sur cette séquence de bits. Le résultat obtenu après ceci doit être égal au code OTP précédent. Si c'est le cas, alors l'utilisateur est authentifié et le code entré par l'utilisateur est stocké de façon à calculer les prochains codes lors des futures authentifications.

Nous avions évoqué ci-dessus que la fonction de hashage est appliquée N fois donnant lieu à un code OTP initial. Nous savons que les codes suivants sont calculés en appliquant N-x fois cette même fonction. Cela veut dire qu'à un certain moment N=0.
Il faudra donc réinitialiser la séquence. Soit le système permet à l'utilisateur de le faire, soit en demandant le premier code OTP de la séquence. Le système devra alors s'assurer que la seed ou la pass-phrase ai(en)t bien été changée(s).

D'un point de vue sécurité, le système OTP est basé sur la fonction de hashage qui est irréversible et quasi impossible à recalculer par un ordinateur. Les codes temporaires rendent les attaques de rejeu très difficiles. De plus, sachant qu'aucun secret ne passe sur le réseau, les attaques de type malwares,

spywares, MITM sont quasiment inutiles.
L'évolution des attaques sur le Web a fait que le code OTP est devenu une méthode d'authentification repandue pour renforcer la sécurité. C'est notamment le cas du monde bancaire et de celui des jeux vidéos. Cette authentification forte allie sécurité et utilisabilité. Il est cependant parfois inadéquat à certaines personnes plus lentes ou ayant des problèmes de compréhension pour ce type de système. Néanmoins, les OTPs ont quand même été largement adoptés par la communauté utilisateur.

2.4.3 Biométrie

La biométrie est un système permettant de reconnaitre et d'authentifier un individu sur base de ces attributs physiques. Il existe plusieurs moyens de reconnaissance mais les plus courantes sont le visage, l'empreinte digitale, la géométrie de la main, l'iris, « keystroke » (taper au clavier), la signature et la voix [Jain et al., 2006]. Néanmoins , les quatre principaux utilisés sont l'empreinte digitale, le scan de l'iris, la reconnaissance faciale du visage et enfin la reconnaissance vocale.
La biométrie est très peu répandue lors de l'authentification Web depuis un ordinateur. Cependant, elle est tout à fait adaptée aux environnements sur smartphone et tablette. Nous distinguons deux étapes lors de l'utilisation d'un système biométrique. Afin de pouvoir authentifier un utilisateur, il faut donc s'assurer que l'attribut physique servant de référence soit lié à son compte dans le système. Cela se fait lors de l'étape d'enregistrement. Lors de celle-ci, l'utilisateur devra utiliser une première fois le système afin de créer sa correspondance et l'enregistrer dans une base de données ou encore sous un device hardware qu'il pourra présenter lors de l'authentification. Cette phase est critique puisqu'un mauvais enregistrement empêcherait toute authentification future auprès du système. L'authentification, quant à elle, est scindée en trois phases distinctes :

- La première est la vérification, autrement dit, est-ce que la personne est vraiment celle qu'elle prétend être ?
- La deuxième est la phase d'identification, dans laquelle, l'application va aller chercher une correspondance entre l'entrée donnée et celle présente dans la base de données.
- La derniére, le screening, consiste à voir à quelle liste l'utilisateur appartient, et ce, afin de lui attribuer des droits spécifiques.

Certaines variations biométriques telles que par exemple un mauvais signal peuvent provoquer des résultats biaisés comme des faux positifs ou négatifs. Ils sont aussi surnommés FTC et FTE.
D'un point de vue sécurité, ces faux positifs et négatifs sont un des désavantages de la biométrie. Une personne non autorisée pourrait avoir accès à certaines choses sans y être autorisée. Cependant, le taux de ceux-ci restent quand même très faible et tout système biométrique est à associé à une authentification forte.
La biométrie peut être sujette à diverses attaques, mais les plus ciblées sont celles qui usurpent l'identité ou qui contourne le système. Comme décrit ci-dessus, un individu pourrait profiter d'un faux positif

pour entrer dans le système. Ou il pourrait encore voler l'attribut physique de la personne[7]. Certaines attaquent se basent aussi sur le fait qu'il soit possible de forger un artefact[Jain et al., 2006] basé sur une image ou photo de la personne afin de faire une attaque de type spoofing à l'encontre du système.
La biométrie a le majeur avantage de nécessiter la présence physique de l'individu sur place. La mise en place d'une PKI permet d'ajouter une couche cryptographique supplémentaire en chiffrant les empreintes lors de l'enregistrement et donc d'augmenter le niveau de sécurité de ce système.
Sachant que la biométrie se base sur les attributs physiques des personnes, la notion de privacy peut se poser. Ceci sera discuté dans le dernier chapitre de ce mémoire.

2.4.4 Certificat

L'authentification par certificat est une méthode d'authentification forte dont la sécurité est basée sur la cryptographie asymétrique. Ceci implique donc la mise en place d'une PKI côté serveur.
Dans un flux d'authentification par certificat, nous distinguons deux acteurs : le client[8] et le serveur.
Pour réaliser une authentification correctement, le serveur va avoir besoin d'informations présentes dans le certificat. Voici les informations que ce dernier comporte :

- Une clé publique.
- Des informations supplémentaires concernant l'entité représentée (nom, prénom, adresse, etc...).
- La signature des autorités le certifiant de confiance.

La clé publique est associée à une entité, humain ou machine.
Il existe trois niveaux de certificats[Colin, 2016]. En fonction du niveau utilisé, le contrôle de l'identité est renforcé.

- "Le premier niveau va permettre de contrôler l'adresse email du demandeur".
- "Le second niveau effectuera un contrôle à distance, en vérifiant typiquement les papiers d'identité, via une photocopie reçue par exemple".
- "Le dernier niveau demandera une présence physique lors de la vérification".

N'importe qui peut créer et signer un certificat. Cependant, afin d'être reconnu comme étant digne de confiance, un certificat doit être signé par une autorité de certification. Cette autorité est elle-même signée par une autorité supérieure ou par elle-même si elle se trouve tout en haut de la chaine de certificats. Nous parlons ici d'une chaine de certification. Il en existe deux types : la chaine dite hiérarchique et celle dite cross-certification.
La première est la plus fréquente. Chaque niveau certifie comme étant de confiance, le niveau en dessous de lui. La seconde permet d'établir des relations de confiance entre différents domaines. Celle-ci est souvent utilisée dans le cadre de partenariat entre sociétés. Si plusieurs domaines certifient un certificat alors il peut être considéré comme étant de confiance.
Un aspect important des certificats est que bien que considérés comme étant de confiance, ils peuvent ne

7. Ceci est extrèmement rare mais reste possible.
8. Au niveau Web, nous parlons de client avec son navigateur.

plus être valides. Soit leur date de péremption est dépassée, soit ils peuvent être invalidés pour d'autres raisons (compromis, révoqués, erronés). Quand c'est le cas, ils se retrouvent dans des listes. Ces dernières sont appelées CRLs. Une CRL est un petit fichier contenant l'ID des certificats qui ne sont plus valides. Elles sont émises et signées par les autorités de certification. A noter que la mise en place et la gestion de ces dernières sont parfois assez contraignantes.
Pour pallier à ceci, les sociétés délivrant les certificats ont mis en place un outil de vérification en ligne : le protocole OCSP. Il est d'ailleurs généralement conseillé de l'utiliser par simplicité.

Lors de la phase d'authentification, nous pouvons voir une session TLS / SSL classique[J.Moore, 2014] .

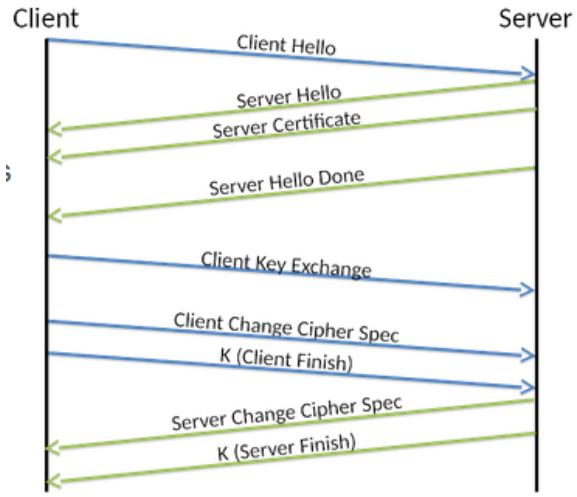

FIGURE 2.1 – Session TLS / SSL

L'authentification par certificat se base sur ce schéma mais ajoute trois étapes :

- Une requête de demande de certificat par le serveur.
- L'envoi du certificat par le navigateur du client.
- La vérification du certificat client. Elle consistera en un hash des messages échangés lors de la session qui sera chiffré avec la clé privée du certificat client. Le serveur va décrypter ce hash et le recalculer. S'il arrive à le recalculer alors le client sera authentifier.

Le schéma ci-dessous montre le flux TLS/SSL avec les trois étapes supplémentaires, citées ci-dessus[J.Moore, 2014].

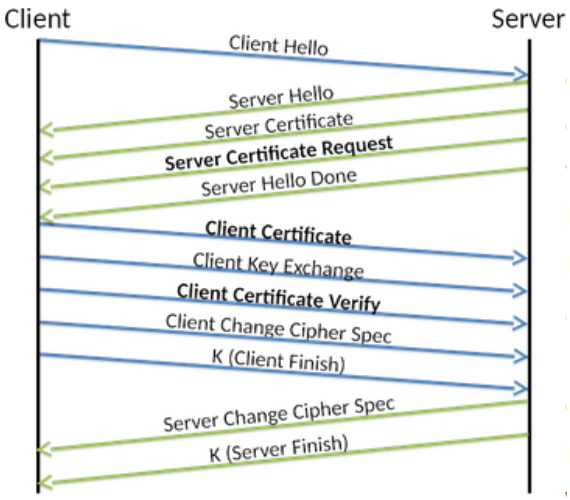

FIGURE 2.2 – Flux d'authentification par certificat

Les certificats possèdent plusieurs qualités de sécurité. Ils garantissent la non-répudiation, la confidentialité via le chiffrement et l'intégrité des données via la signature numérique. De plus, une authentification par certificat permet d'éviter qu'un secret transite sur le réseau. Les certificats sont utilisés dans tous les secteurs depuis de nombreuses années. Ils sont donc une technologie mature. Leur utilisation nécessite parfois une formation des utilisateurs.

2.4.4.1 Les cartes d'identité electroniques

Une des implémentations basées sur les certificats est la carte d'identité ou carte eID. De plus en plus de pays ont adopté ce système plutôt que la gestion papier. Cela comporte plusieurs avantanges. Premièrement, une économie de papiers, ainsi qu'une centralisation des données citoyennes beaucoup plus aisée. Ensuite, l'aspect non-répudiation est renforcé par le fait qu'il soit possible de connaitre le nom de la personne physique effectuant telle ou telle action sur le site. De plus, la carte contient beaucoup de données concernant le citoyen, ce qui facilite les taches administratives, les inscriptions, les cartes de fidélité etc... Cependant, un des gros désantavages de ce système est la protection de ces données privées. Les sites récoltant celles-ci ne les utilisent pas toujours en bon père de famille, comme précisé dans cet article[Bichsel et al., 2009]. Les sites bancaires, de commerce en ligne ou encore gouvernementaux sont friands de ce type de système. La non-répudiation dans une transaction est essentielle pour ces organismes. La personne ne pourra nier qu'elle a effectué l'action. Or, c'est une qualité indispensable pour eux.

2.4.4.2 L'eID : le modèle belge

La Belgique ainsi que l'Autriche sont dans les premiers pays à avoir utilisé la carte d'identité lors de transactions électroniques. Nous allons expliquer le système belge car il nous concerne. Le système autrichien reste disponible en annexe L'eID : le modèle autrichien. La carte d'identité est de la taille d'une carte bancaire et contient différentes informations personnelles relatives au citoyen. En Belgique, ces principales informations sont le nom, le(s) prénom(s), le lieu et la date de naissance , l'adresse ainsi que le numéro de registre national. Ce numéro unique est lié à la personne. La date de validité de la carte

est aussi présente. Toutes ces informations sont non seulement visibles sur la carte mais elles sont aussi présentes dans un fichier situé sur le chip de la carte. La taille de ce fichier est très petite puisqu'il fait à peu près deux cent bytes. Ce fichier est signé par le RRN. Plusieurs utilisations de la carte sont possibles puisqu'elle contient trois clés privées de 1024 bits[9] RSA permettant d'effectuer l'authentification ou des signatures digitales différentes[Danny De Cock and Preneel, 2005] :

1. Une clé servant à l'authentification du propriétaire de la carte. Nous parlons ici d'une authentification classique sur un site Web proposant l'authentification par carte eID.
2. Une clé servant à faire des signatures de façon à s'assurer de la non-répudiation lors d'une signature digitale. Nous pouvons par exemple signer un contrat électroniquement de cette façon.
3. Une clé servant à authentifier la carte auprès des systèmes du gouvernement belge.

Lors de l'utilisation des deux premières clés, un code PIN va être demandé à l'utilisateur. Un certificat est associé à chacune de ses deux clés. En outre, chaque paire de clés possède un certificat lui étant associé. Le premier servira lors d'une authentification client-serveur. Le deuxième, quant à lui, servira de liaison entre le propriétaire de la carte et la non-répudiation lors d'une transaction électronique. Le troisième sera utilisé afin faire une authentification mutuelle entre le RRN et la carte ainsi que lors de mise à jour de données de celle-ci. Chacun de ces trois certificats se voit lié à une autorité qui les certifie. Les voici :

1. Le certificat Belgian Root CA.
2. Le citizen CA.
3. Le certificat RRN.

L'image ci-dessous montre la chaine de certification utilisée lors d'une authentification par carte d'identité.

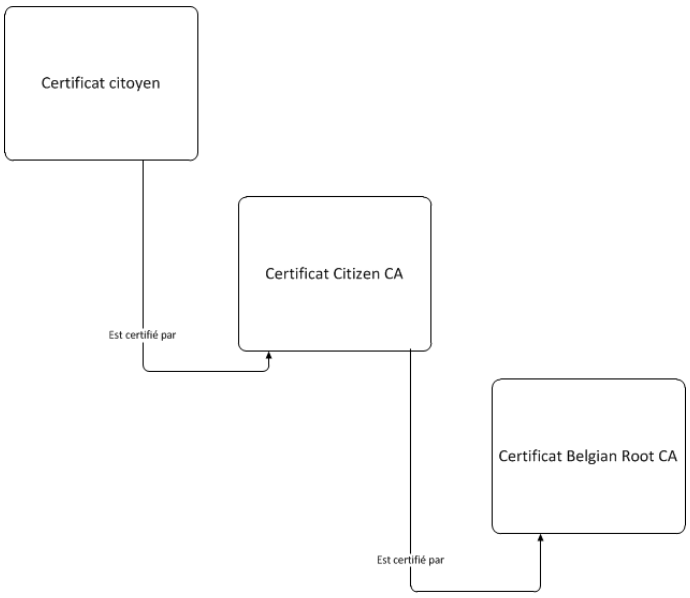

FIGURE 2.3 – Chaine de certification

9. Nous parlons de 1024 bits dans la documentation lue. Il est fort probable que les nouvelles clés soient en 2048 bits.

Lors de la vérification de cette chaine, si un des maillons n'est pas correct, l'utilisateur sera automatiquement rejeté. De la même façon que pour les certificats classiques, la validité des certificats des cartes d'identité est gérée via les CRLs ou le protocole OCSP. Ces CRLs sont généralement délivrées chaque mois sur les sites officiels. Néanmoins, il existe des delta-CRL qui sont émises plus régulièrement. Elles permettent de mettre à jour les bases de données de certificats non utilisables plus rapidement.
Les systèmes actuels se basent de plus en plus sur les attributs contenus dans les cartes eID afin d'identifier et d'assigner un rôle à l'utilisateur. Nous remarquons que le protocole SAML est un bon candidat pour ce type de gestion. L'avantage de cette association est un renforcement supplémentaire de la sécurité . Nous présenterons le système développé par le Commission Européenne plus tard dans ce document. Il est entièrement basé sur l'association que nous venons de citer.

2.4.5 Authentification graphique

L'authentification graphique est une méthode d'authentification qui utilise des mots de passe graphiques afin de laisser accéder les utilisateurs aux zones privées des applications.
Ce système est basé sur la reconnaissance d'images. Il implique deux acteurs : le client avec son navigateur et le serveur. Son principe est de prouver au serveur que nous connaissons le secret, sans pour autant le divulguer.
Dans un flux complet, nous distinguons deux phases : l'enregistrement et l'authentification.
Durant la phase d'enregistrement, l'utilisateur va choisir un pattern d'images ou choisir des points dans une image. Ce choix représentera l'équivalent de la connaissance du secret.
Dans la suivante, il devra s'authentifier avec le pattern créé précédemment. En fonction du type d'authentification, l'utilisateur devra rejouer celui-ci.
Nous parlons du type d'authentification graphique car il en existe trois :

- Le premier est la reconnaissance. Ce dernier utilise la reconnaissance sur base d'un visage humain. L'outil PassFace, décrit dans cet article[shyong Tsai et al., 2006], utilise par exemple ce principe.
- Le deuxième est le rappel, comme l'outil commercialisé GrIDSure[Wikipedia, 2015]. Ce système est basé sur une grille. A l'enregistrement, cette grille est affichée et l'utilisateur va choisir un pattern. Lors de l'authentification, cette même grille sera présentée mais cette fois, elle sera remplie de chiffres mis aléatoirement dedans. L'utilisateur sera invité à donner les chiffres représentant le pattern choisi à l'enregistrement.
- Le troisième et dernier système est l'indice-rappel. A l'enregistrement, une image sera présentée et l'utilisateur devra choisir des points sur celle-ci. A l'authentification, cette même image sera affichée et il devra donner l'emplacement des points choisis, moyennant une marge ou zone d'erreur. Ce système est utilisé par PassPoints.

La sécurité de cette méthode d'authentification est plus forte que celle des mots de passe. Elle permet d'éviter des attaques comme le phishing, les spywares et certains types de malwares [10] Cependant, elle est quand même très fortement faillible à une attaque en particulier, le shoulder-surfing[shyong Tsai et al., 2006].

10. Il existe des malwares prenant des captures d'écran.

Il est très simple pour un utilisateur malveillant de regarder le pattern lorsqu'il est rejoué. Sachant que le pattern est généralement constitué de cinq points ou images à reconnaitre, il est simple à retenir.

En pratique, les systèmes d'authentification graphique sont très peu utilisés sur le Web, probablement de par leur manque de maturité. Cependant, nous remarquons qu'ils sont beaucoup utilisés par les mobile devices, lors de l'accès principal à l'interface du smartphone.

2.4.6 Token Hardware

Le principe de ces tokens se base sur un device externe qui est utilisé comme premier ou second facteur d'authentification. Par exemple, le PC-Banking utilise un lecteur de carte dans lequel l'utilisateur entre un code donné par la banque et son code PIN de la carte pour ensuite recevoir un autre code à entrer sur le site. Tout ceci est basé sur des codes OTP. Certains autres systèmes vont envoyer un code OTP via d'autres canaux, comme le SMS ou l'email. C'est notamment le cas de la société Blizzard[Blizzard, 2015], qui propose ce système pour pallier aux vols de compte utilisateur, assumant donc que tout utilisateur possède au moins un téléphone portable. Certains types de VPN utilisent des codes OTP afin de renforcer l'authentification et donc l'accès au réseau privé de l'entreprise. La société Yubico propose quant à elle, une clé USB, la Yubikey. Il existe d'autres systèmes similaires, comme RSA SecurID ou encore IronKey. Ces derniers vont être expliqués ci-dessous.

2.4.6.1 Yubico

La Yubikey[Yubico, 2016], développée par la société Yubico est un token hardware de type stick USB. Elle est utilisée lors d'une authentification multi-facteurs. Grâce à ce système, les attaques de type phishing, malware ou spyware sont peu efficaces. Yubico se base sur les principes U2F de FIDO, qui seront expliqués en section 2.5.5.
Son utilisation se fait en deux phases :

- La première phase est l'enregistrement, dans laquelle l'utilisateur va enregistrer sa clé Yubikey auprès du service.
- Dans la seconde phase, l'utilisateur va s'authentifier de façon classique via un login et un mot de passe. Après cela, le service va vouloir s'assurer que l'utilisateur possède la clé et va lui demander via un challenge-response. Si la clé est insérée dans un port USB, un simple touché sur le bouton va permettre d'envoyer l'information nécessaire au service.
  Le diagramme de séquence ci-dessous illustre cette phase :

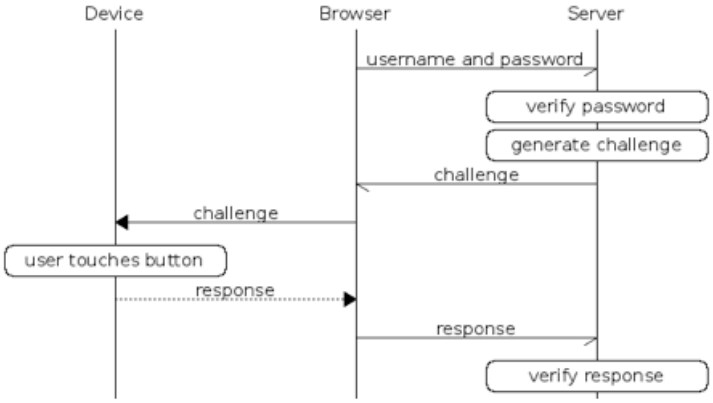

FIGURE 2.4 – Authentification renforcée avec la clé Yubikey[Yubico, 2016]

Ce token assure la sécurité et une utilisabilité correcte pour l'utilisateur.
Du point de vue de la sécurité, la clé peut être utilisée dans différents contextes :

1. Dans le cadre des Password Manager, elle permet de renforcer la sécurité du mot de passe principal.
2. Les services Internet deviennent de plus en plus la cible d'attaque, notamment les sites Gmail, GitHub, Dropbox etc... ELle permet de renforcer l'authentification vers ceux-ci.
3. La clé peut aussi intervenir lors de l'ouverture d'une session sur une machine dont le système d'exploitation est Windows, Linux ou Mac.
4. La clé permet aussi de renforcer le chiffrement de données sur un disque.

2.4.6.2 RSA SecurID

Le token RSA SecurID [RSA, 2015], souvent appelé 'Dongle' dans le jargon des utilisateurs, est utilisé dans le cadre d'un renforcement de sécurité lors d'une authentification par mot de passe. Par rapport à un système de type token USB, aucune connexion physique ne doit être établie avec la machine cliente. Comme la plupart des tokens, le fonctionnement se fait en deux phases : l'enregistrement et l'authentification.

- L'enregistrement est une phase classique où le service est notifié du renforcement de sécurité via un second facteur d'authentification.
- La phase d'authentification, quant à elle, est un peu différente. L'utilisateur va entrer son mot de passe de façon classique. Cependant, il lui a été fourni un code PIN, qu'il connait. Il devra entre ce code PIN ainsi que le code OTP qui sera affiché sur son doggle lors de cette phase.

L'image ci-dessous représente un des tokens hardware RSA. Il en existe d'autres :

FIGURE 2.5 – Token RSA

Les tokens RSA sont principalement utilisés dans le cadre d'accès VPN ou extranet. Cependant, ils peuvent être utilisés dans d'autres cas tels que l'authentification sur à un Cloud, le renforcement de la sécurité des mobile devices de type smartphone ou encore lors d'une politique de type BYOD.

2.4.6.3 IronKey

La clé USB , IronKey [IronKey, 2015] est du même type que la Yubikey. Elle est utilisée lors d'une authentification multi-facteurs. Cependant, elle diffère quelque peu puisque le second facteur est une authentification biométrique via la clé. Un simple passage devant la clé avec le doigt permet d'authentifier l'utilisateur auprès du service.

FIGURE 2.6 – Token Ironkey

L'Ironkey peut aussi être utilisée dans d'autres cas.

1. La clé permet de renforcer l'authentification lors de l'accès à une machine cliente.
2. La clé permet de renforcer le chiffrement de données sur un disque dur ou un disque flash.
3. La clé peut être utilisée dans le cadre d'une politique BYOD.
4. La clé permet le renforcement d'accès à des machines distantes de type VDI ou des machines partagées.

L'avantage d'un point de vue de la sécurité est que non seulement il faut le token mais aussi la présence physique du propriétaire. En plus de la sécurité renforcée par le multi-facteurs, le vol de la clé est complètement inutile [11].
Les différents secteurs d'utilisation sont les secteurs de la santé, gouvernemental, fédéral, éducation ou encore financier.

11. Partant du principe que l'attaquant ne possède pas l'empreinte digitale électronique du propriétaire de la clé.

2.4.7 Authentification multi-facteurs

L'authentification multi-facteurs permet de combiner plusieurs méthodes d'authentifications différentes. Généralement, elle se compose d'un mot de passe en premier facteur et d'un code OTP ou d'un certificat en second facteur. Cependant les facteurs d'authentification ne sont pas figés. Nous pourrions avoir un premier facteur qui est un certificat et un second qui est un facteur biométrique.
Ce système est fortement répandu sur les sites Web nécessitant une sécurité renforcée. Nous parlons typiquement de deux facteurs d'authentification , mais il pourrait y en avoir plus. Il est possible de chainer les facteurs avec différentes méthodes. TaxOnWeb[Belgium.be, 2015] est un exemple utilisé par un grand nombre de citoyens en Belgique. Peu s'en rendent compte mais ils utilisent une authentification beaucoup plus sécurisée via ce système. TaxOnWeb propose deux choix de multi-facteurs :

- Soit un choix où le premier facteur demande un mot de passe classique avec un second facteur qui est un OTP sous forme d'une liste de codes détenues par le citoyen.
- Soit un autre choix avec la carte d'identité, impliquant un certificat ainsi qu'un code PIN.

Avec la croissance du Web et des business autour de celui, les authentifications multi-facteurs sont devenues largement utilisées par tout type de société, publiques comme privées. C'est notamment le cas des sites gouvernementaux, qui commencent à prendre conscience des avantages qui y sont liés. Ceci en fait un atout dont la maturité et la sécurité ne sont plus à prouver. Malgré cela, nous remarquons que certains utilisateurs restent en retrait par rapport à ce système. Ceci est dû au fait que l'utilisabilité de l'authentification multi-facteurs demande parfois une formation de l'utilisateur et du fait qu'il ne comprenne pas toujours l'engouement pour ces différentes couches de sécurité.

2.4.7.1 Google Authenticator

Google Authenticator est un système d'authentification développé par Google. L'authentification est basée sur des codes QR, c'est à dire des codes barres à scanner avec un smartphone[Google, 2015]. Lors de l'initialisation, un code QR sera présenté afin d'être scanné. Il permettra de générer une clé partagée entre le serveur et le smartphone. Lors de l'authentification, un nouveau code QR temporaire sera présenté, il sera une combinaison entre la date de l'horloge et la clé enregistrée lors de l'initialisation. Sachant que le serveur et le smartphone sont normalement réglés à la même heure, vu qu'ils sont synchronisés via internet, ils pourront exécuter le même algorithme, permetant de vérifier que les résultats sont semblables des deux côtés.
L'avantage principal d'un point de vue sécurité est que ce système permet une authentification forte, à deux facteurs. Il permet de renforcer la sécurité des comptes Gmail, Dropbox ou encore des accès SSH sur une machine Unix.

2.5 Les protocoles d'authentification

2.5.1 Password Manager

Les Password Manager sont des outils permettant de gérer l'ensemble des mots de passe de l'utilisateur. Afin de récolter ceux-ci pour des applications Web, un plugin est nécessaire au niveau du navigateur comme Sync[Mozilla, 2015] de Mozilla ou LastPass[Gibson, 2016]. A chaque première visite d'un site ou d'une application, le plugin va enregistrer les identifiants de l'utilisateur. Par la suite, quand l'utilisateur va se rendre à nouveau sur l'application, le Password Manager va préremplir automatiquement les champs avec les identifiants. En fonction de la solution choisie, les Password Managers offrent différentes fonctionnalités telles que le chiffrement, hashage, sauvegarde dans le cloud, mot de passe principal, mise à jour et récupération des mots de passe.
Toutes ces fonctionnalités sont néanmoins un vecteur d'attaque supplémentaire. Cela est particulièrement le cas de l'implementation du mot de passe principal. Si ce dernier est compromis, un attaquant aura alors accès à toutes les applications. Il est notammement utilisé avec les mobiles device mais est considéré comme peu sécurisé[Belenko and Sklyarov, 2012].

Il existe une multitude d'attaques concernant les Password Manager[Silver et al., 2014]. Les principales sont les 'sweep attacks' faisant de l'extraction de données ou encore les 'attaques de type injection' (XSS, injection dans la page de login,..).
L'idée derrière les Password Manager reste de la gestion de mot de passe à travers plusieurs services. De par leur fonctionnement, ils sont considérés comme un SSO local sur la machine de l'utilisateur.

2.5.2 Proxy-based

Le proxy est un service sur lequel l'utilisateur s'authentifie et qui ensuite, va transmettre le statut de l'authentification ou rejouer la méthode auprès du service demandé. Ce dernier va se situer entre la machine cliente et l'application. Il n'est pas lié à une méthode d'authentification en particulier. Un utilisateur peut s'authentifier sur le proxy via mot de passe, OTP, certificat ou encore une méthode de type multi-facteurs. Le terme 'proxy' n'est pas limité aux technologies proxy connues mais il peut aussi représenter une application tierce effectuant une authentification, comme les IdPs. Le proxy part du principe que la machine cliente n'est pas sécurisée et qu'en s'authentifiant par son biais, la sécurité ne sera pas compromise entre lui et l'application. Cela peut se faire via des politiques de sécurité au niveau de ce dernier.
Il existe trois types de familles de proxies : HTTP, MITM et SSO.
Les proxies HTTP comme KLASSP[Florencio and Herley, 2006] ou Impostor[Andreas Pashalidis, 2004] nécessitent une configuration du navigateur lors de l'utilisation tandis que les proxies MITM[Florêncio and Herley, 2008] ou URRSA[Jammalamadaka et al., 2006] sont transparents pour l'utilisateur. Impostor est un proxy de HTTP mais aussi un proxy SSO. Néanmoins, ces quatres restent des proxies développés dans le cadre de la recherche et ne sont pas utilisés par la communauté des utilisateurs. Les proxies SSO sont géné-

ralement des applications tierces qui agissent en tant qu'IdP. Ces derniers sont monnaie courante dans le cadre de l'authentification Web puisqu'ils permettent d'implémenter l'authentification unique auprès de n'importe quelle d'application. Cela permet une politique de sécurité concentrée au niveau du proxy, tant en assurant un SSO inter-applications.
Un des désavantages des proxies est qu'ils sont des cibles privilégées par des hackeurs. Cependant, en cas de compromission, l'application n'est pas réellement compromise. Une coupure du lien entre l'application et le proxy suffit à limiter les dégats. Les administrateurs systèmes peuvent alors associer un autre proxy ou un autre système d'authentification à l'application.

2.5.3 Phone-based

Le principe de ce système est basé sur deux composants , un mobile device de confiance et une machine publique, c'est à dire qui n'est pas confiance. Le but est de pouvoir s'authentifier de la machine publique de façon sécurisée, et ce, grâce au mobile device. Ce dernier va avoir le rôle d'intermédiaire entre la machine publique et le service. Phoolproof[Parno et al., 2006] est par exemple basé sur une authentification SSL entre le navigateur et le serveur via le mobile device. Tandis que MP-Auth[Mohammad Mannan, 2007] va baser son système sur la transformation d'un mot de passe en quelque chose d'inutile pour un attaquant, un one-time password. Ceci est fait sans que le mot de passe soit révélé. Les systèmes basés sont les mobiles device sont résistants à la plupart des attaques de type phishing ou spyware mais la seule contrainte est de s'assurer que le mobile device soit sécurisé. Il est donc le point critique de ce système.
Les systèmes basés sur les smartphones fonctionnent très bien dans un cadre Web. Phoolproof et MP-Auth ont été développés dans un cadre de recherche cependant leur principe reste intéressant. Nous remarquons qu'avec l'expansion des smartphones, l'authentification via un mobile sécurisé pourrait devenir de plus en plus répandue.

2.5.4 Le SSO

Le but du SSO est de permettre l'authentification unique. Ce qui veut dire que l'utilisateur ne s'authentifie qu'une seule fois à un seul endroit et a la possibiité d'accéder à un certain nombre de services sans devoir se ré-authentifier à chaque fois. Généralement, l'authentification principale du SSO est un mot de passe cependant, ceci n'est figé. Le SSO est totalement indépendant de la méthode d'authentification utilisée.
Beaucoup d'articles scientifiques tels que ceux de Sun[Sun et al., 2011] et de Florencio[Florencio and Herley, 2007] rapportent qu'un "utilisateur lambda possèdent en moyenne 25 comptes et 8 mots de passe différents pour les gérer". Ils indiquent aussi que cette tendance ne va pas diminuer avec l'accroissement du nombre d'applications disponibles sur le Web. De plus, par facilité, les utilisateurs choisissent les mots de passe à faible entropie.
Les chercheurs ont démontré "qu'il existait une certaine lassitude à retenir des mots de passe différents". Les utilisateurs veulent un système facilement utilisable, entrainant le moins d'actions possibles. Le fait de retenir une multitude de mots de passe crée une certaine fatigue. Ceci explique pourquoi ils ne possèdent

que si peu de mots de passe pour autant de sites Web. La conséquence de ces faits est que la sécurité en est affectée.

Le SSO a donc été implémenté dans le but de pallier à cette fatigue de mémorisation.
Il faut quand même noter que le SSO possède des avantages d'un point de la sécurité.

- Premièrement, le fait de ne s'authentifier qu'une seule fois diminue le nombre de fois où un utilisateur doit entrer son mot de passe et donc le nombre de possibilités de compromission de ceux-ci. Moins un utilisateur tape son mot de passe, moins des attaques de type spywares, malwares, keyloggers etc... fonctionneront.
- Deuxièmement, en partant du principe que la méthode d'authentification principale du SSO est un mot de passe, les administrateurs peuvent appliquer facilement des règles de sécurité sur ceux-ci. Les règles les plus connues sont souvent un nombre de caractères minimum et maximum, une majuscule et un chiffre dans le mot de passe ou encore changer ce dernier tous les x semaines.
- Une troisème avantage, est de pouvoir gérer les sessions plus facilement et par exemple d'appliquer un timeout de façon globale.
- Quatrièmement, il est facile de retracer une identité puisqu'elle sera associée au même utilisateur à travers l'ensemble des applications.

Du point de vue de la sécurité, le désavantage principal du SSO est le même que celui des proxies : la compromission de l'authentification principale.
Parmis les systèmes de SSO, il existe deux grandes catégories : les Pseudo-SSO et les True-SSO. Ces deux catégories sont chacune divisées en deux sous-ensembles, les Proxy-based SSO et les Local SSO.

2.5.4.1 Les Pseudo-SSO

Les Pseudo-SSO sont un des deux grands types de SSO. Leur principe est d'exécuter le mécanisme d'authentification auprès des différents SPs connus. Ils vont donc se situer entre les SPs et le navigateur du client, agissant comme un proxy.
Afin de connaitre les différents SPs, ceux-ci doivent être enregistrés auprès du composant SSO. Pour chaque SP, cela comprend généralement son URL et la méthode d'authentification qui lui est liée. Peu importe le mécanisme d'authentification (mot de passe , certificat, etc...), le Pseudo-SSO réalisera la phase d'authentification. Lors de cette phase, l'utilisateur va d'abord s'authentifier sur le composant SSO et choisir le service auquel il voudrait accéder. Lors de la première authentification sur le service, un mécanisme telle qu'une base de données ou un fichier plat sera mis en place pour garder en mémoire la façon de se connecter au service. Cela inclut donc les identifiants, certificats ou tout autre donnée relative à l'utilisateur qui lui serait utile. Lors des authentifications suivantes, le Pseudo-SSO rejouera l'authentification de façon transparente pour l'utilisateur. Il est évident que si du côté du service, il y a un changement dans la méthode d'authentification ou une mise à jour quelconque des identifiants, ceci devra être répercuté au niveau du composant Pseudo-SSO.

Afin de conserver la session entre la machine client et le composant Pseudo-SSO , il sera généralement mis en place un mécanisme tel que la création de cookie. Ceci implique donc que le navigateur client doit les accepter, sinon l'utilisateur devra se re-authentifier à chaque appel. Une fois le navigateur fermé ou le cookie détruit dans le navigateur, le composant Pseudo-SSO détectera que le cookie n'est plus présent et invalidera la session côté serveur. La session au niveau du service devra être gérée entièrement par lui-même, le composant n'intervient en rien la dedans. Chaque accès a un nouveau service nécessitera une authentification, réalisée par le composant Pseudo-SSO.
D'un point de vue identité et autorisation, chaque utilisateur peut posséder plusieurs identités au sein des services. Les composants Pseudo-SSO ne gèrent que de façon limitée les identités au sein des applications. La gestion des identités des utilisateurs est généralement laissée au sein de ces dernières.
Le schéma ci-dessous montre l'authentification via un composant pseudo-SSO :

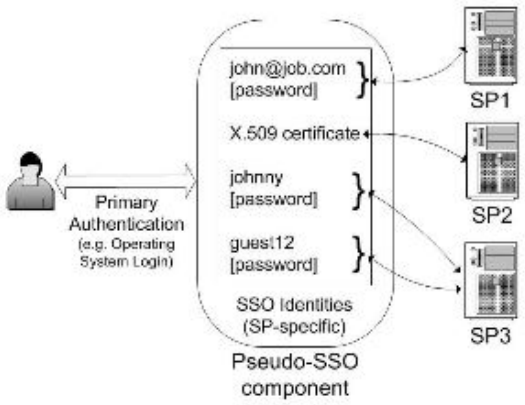

FIGURE 2.7 – Pseudo-SSO[Pashalidis and Mitchell, 2003]

Un exemple de composant de ce type qui prend de l'ampleur est le Web Access Manager développé par Evidian[Evidian, 2016].
La sécurité des Pseudo-SSO peut être comprise par plusieurs types d'attaque. Premièrement, comme tous les SSO, il y a un point d'entrée principe qui est une cible pour tout attaquant. Tout malware, spywares ou autre attaque volant les identifiants principaux peuvent faire beaucoup de dégats. Deuxièmement, sachant que les composants pseudo-SSO exécutent la phase d'authentification auprès des SPs, ils sont subceptibles aux attaques de rejeu , comme le fait remarquer Kothari[Kothari, 1985]. Il recommande de bien sécuriser le point d'entrée du SSO via les protocoles TLS et SSL.

2.5.4.2 Les True-SSO

Contrairement aux Pseudo-SSO, ils n'exécutent pas de mécanisme d'authentification des SPs visés. Ils utilisent des méchanismes particuliers comme des assertions, des tickets ou encore des cookies.
Il y a différents composants liés au True-SSO. L'IdP ou ASP est le composant vers lequel l'utilisateur est redirigé lorsqu'il tente d'accéder à un SP. S'il s'authentifie, ce dernier va alors créer un module prouvant que l'utilisateur est bien authentifié. Celui-ci est alors envoyé SP qui pourra l'interprêter. Il comprend les informations nécessaires permettant au SP de vérifier le statut de l'authentification ainsi que les droits

associés. L'IdP permet de faire une gestion centralisée des droits des utilisateurs à travers les différents SPs associés au composant de True-SSO. Ceci a l'avantage de ne gérer les droits qu'à un seul endroit. Mais en cas de compromission, ce sont tous les SPs qui sont impactés. Les True-SSO impliquent une relation de confiance avec les SPs. Cela peut se faire par des canaux sécurisés via la mise en place d'une PKI.

Mayer[Mayer et al., 2014] propose un schéma de haut niveau montrant l'interaction entre composants du système. Il se situe ci-dessous :

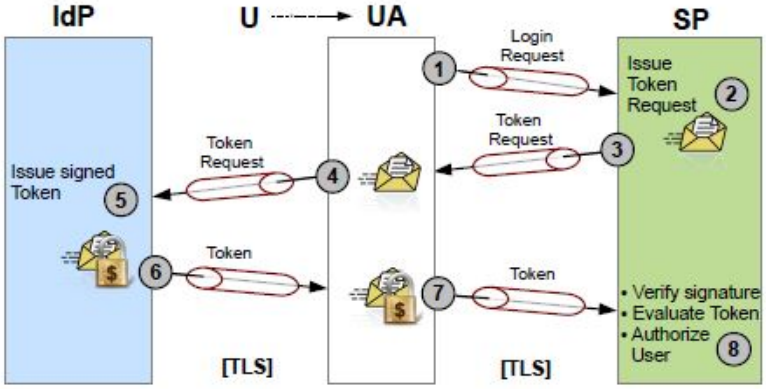

FIGURE 2.8 – Vue haut niveau des True-SSO

Trois acteurs sont présents sur ce schéma : l'IdP, l'UA et le SP. L'UA représente l'utilisateur et son navigateur.

En (**1**), l'utilisateur via son navigateur va tenter d'accéder à une partie du SP qui nécessite une authentification. Le SP (**2**) va alors envoyer un request token au navigateur (**3**), qui lui va le transférer à l'IdP (**4** et (**5**)). Après analyse et vérification du token reçu , l'IdP va alors générer un token pour l'utilisateur et va signer ce dernier. Ceci garantira l'intégrité et l'authenticité. Il répondra au navigateur en lui renvoyant le token créé (**6**). Ce dernier va alors le forwarder vers le SP (**7**) qui va procéder à la vérification de la signature (**8**). Si tout est correct, l'accès sera validé pour l'utilisateur. Tout ce flow se fait généralement de manière sécurisée via TLS, comme indiqué sur l'image.

2.5.4.3 Les SSO locaux

Lors d'un SSO local, le composant va se trouver sur la machine de l'utilisateur. Si c'est un Pseudo-SSO alors une base de données contenant les identifiants chiffrés[12] sera présente sur la machine. Au début de la session, l'utilisateur va s'authentifier sur le composant SSO et ensuite, ce dernier va exécuter la phase d'authentification des services qui seront accédés. La machine doit bien sur être de confiance et non compromise. Si c'est true-SSO, alors une PKI devra être mise en place en faisant attention à l'intégrité et à la fiabilité de la machine.

12. Bonne pratique.

2.5.4.4 Les proxy-based SSO

D'un point de vue Proxy-based, l'infrastructure nécessite un serveur contenant le proxy. Ce serveur doit bien sur être de confiance. Si c'est un Pseudo-SSO , il exécutera son mécanisme habituel lors de la phase d'authentification. En contre-partie, si c'est un True-SSO, alors le proxy aura le rôle de l'IdP en envoyant les informations d'authentification adéquates aux différents SPs.

2.5.5 FIDO Alliance

FIDO Alliance est une société qui a été créée en 2012 par l'initiative de grandes multinationales comme Google et Paypal . Grâce à cette initiative, un ensemble de spécifications ont été développées et de là, ont découlé deux protocoles distincts : UAF et U2F[Alliance, 2014].
Leur but est de trouver une solution au problème des mots de passe, tout en apportant deux avantages majeurs : l'utilisabilité et la sécurité.
Ce système est réalisé via deux composants : le FIDO Server et le FIDO Authenticator[Lindemann, 2014].
Le FIDO Server se situe du côté du service (ou RP) que l'utilisateur tente d'accéder tandis que le FIDO Authenticator du côté du client, c'est à dire de l'utilisateur.

2.5.5.1 Composants et architecture

FIDO Authenticator est souvent associé à un device que l'utilisateur possède et qui lui sert à s'authentifier. Cependant, ce terme désigne plus un concept qu'un device. En effet, "FIDO Authenticator peut représenter un composant software qui tourne sur le FIDO User Device de l'utilisateur. Il peut aussi être implémenté en tant que token hardware dédié comme par exemple une carte ou une clé USB ou encore un software qui tourne dans un environnement de confiance". Cette liste d'implémentation n'est pas exhaustive mais elle explique simplement l'idée derrière ce concept.
FIDO Authenticator peut implémenter n'importe méthode d'authentification, en fonction du contexte voulu. Cela peut être réalisé par le fait que FIDO fait une séparation en protocole d'authentification et méthode d'authentification.
Le protocole utilisé par FIDO lors d'une transaction est l'OSTP. Il est divisé en quatre fonctionnalités[Lindemann, 2014] :

- 'Discovery' permet au RP de comprendre la méthode d'authentification qui sera utilisée par le FIDO Authenticator.
- 'Registration' permet de lier le FIDO Authenticator à une entité. Cela peut être un utilisateur existant ou un nouveau qui sera créé.
- 'Authentication' réalise la création d'un canal d'authentification entre le navigateur et le serveur Web du RP.
- 'Transaction' permet à l'utilisateur d'avoir la vue sur la transaction courante et de l'authentifier au niveau du RP.

L'image ci-dessous représente l'architecture entre un client avec le FIDO Authenticator et un RP sur lequel se situe le FIDO Server.

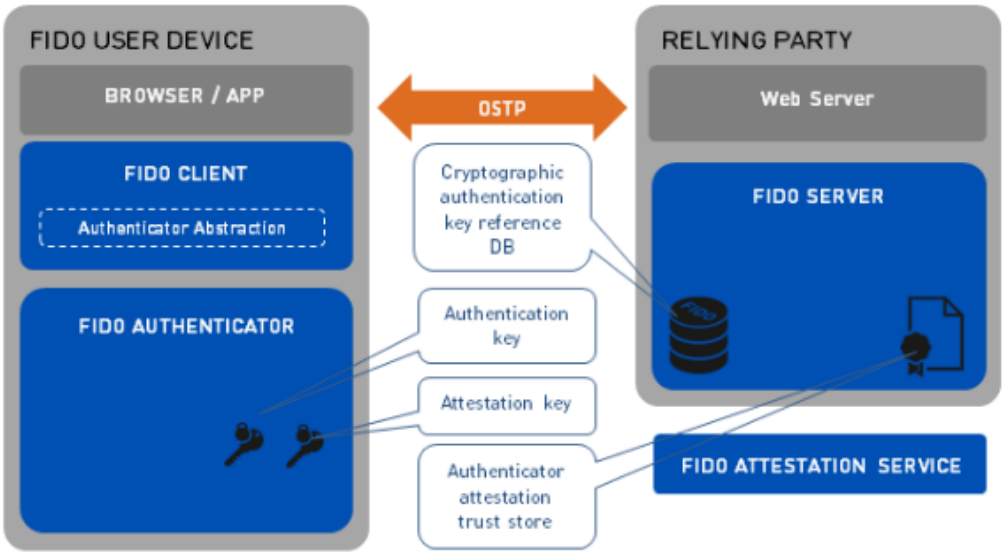

FIGURE 2.9 – Architecture FIDO[Lindemann, 2014]

Ce schéma indique les différents composants présents dans une architecture FIDO client-serveur. Lors d'une authentification de type OTP, biométrie ou autre, un individu malveillant pourrait tout de même interférer sans que le serveur ne fasse la différence. FIDO résout ce problème grâce aux éléments 'attestation key' et 'authenticator attestation trust store'. L'attestation key est la preuve cryptographique que l'authentification est réalisée via le FIDO Authenticator possédé par l'utilisateur. L'enregistrement et l'identification , couplés avec la preuve d'attestation sont présents dans le même certificat. La vérification du côté du serveur est donc unique et sécurisée. Nous retrouvons donc une authentification sécurisé côté serveur et un élément sécurisé côté client.

2.5.5.2 Les protocoles UAF et U2F

Le premier protocole s'appelle UAF. Il permet de remplacer les mots de passe par une authentification forte et/ou d'introduire une authentification multi-facteurs.
Nous pourrions voir une première authentification de type biométrique avec une seconde de type certificat. Son fonctionnement se fait en deux phases : l'enregistrement et l'authentification.

- Lors de la première phase, l'utilisateur enregistre son device auprès RP en sélectionnant la méthode d'authentification locale avec laquelle il veut s'authentifier (PIN, empreinte digitale, parler au micro, etc..). Le RP se réserve le droit de proposer les méthodes voulues par rapport à sa politique d'authentification. Le schéma ci-dessous illustre ceci :

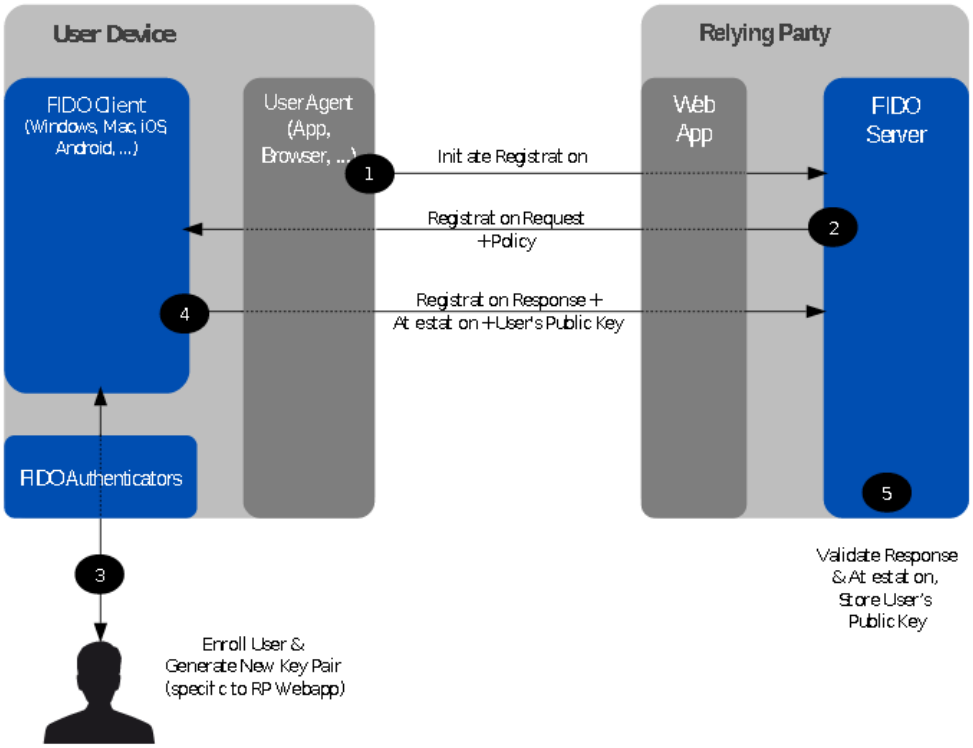

FIGURE 2.10 – Enregistrement UAF[Salah Machani, 2014]

L'utilisateur ayant reçu son Authenticator va faire savoir au RP qu'il veut l'enregistrer(**1**). Le RP va alors envoyer une requête d'enregistrement(**2**), invitant l'utilisateur à jouer l'authentification localement dessus dans le but de le débloquer. Une fois ceci fait, une paire de clés sera générée(**3**). La clé privée restera au niveau de l'Authenticator, tandis que la clé publique sera envoyée et stockée au niveau du RP(**4**). Ce dernier validera la réponse reçue et l'attestation puis liera la clé à l'utilisateur(**5**).

- Lors de l'authentification, l'utilisateur devra rejouer l'authentification qu'il a choisie lors de l'enregistrement. Le schéma ci-dessous illustre ceci :

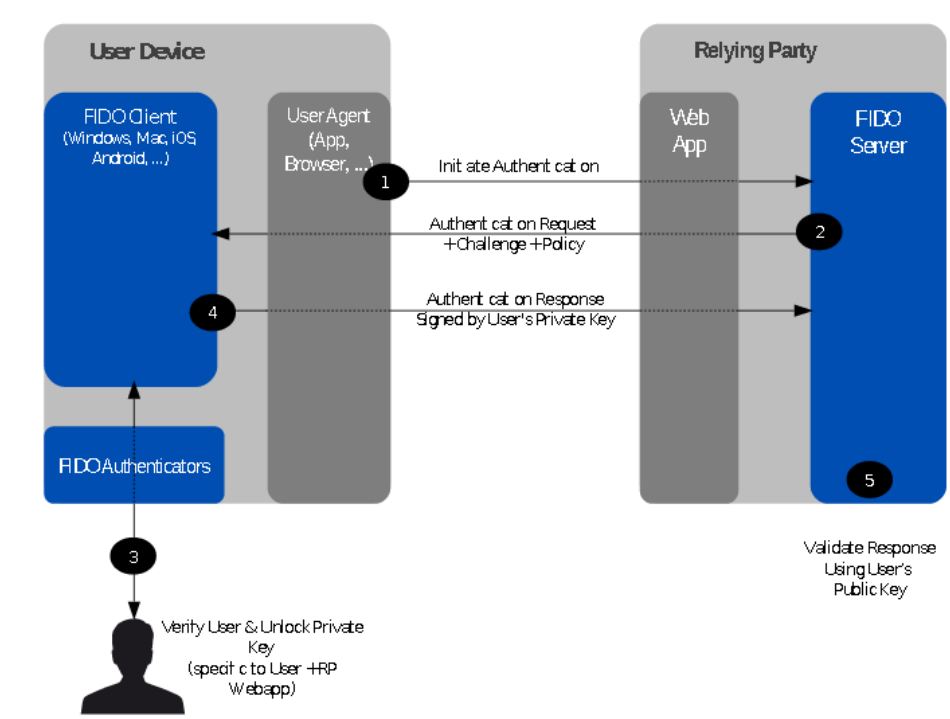

FIGURE 2.11 – Authentification UAF[Salah Machani, 2014]

L'utilisateur va initier la phase d'authentification(**1**). Le RP lui répondre via un challenge-response(**2**). L'utilisateur va s'authentifier localement(**3**) et débloquer sa clé pour signer la réponse. Une fois ceci fait, le client va renvoyer la réponse signée(**4**). Cette dernière sera vérifiée par le serveur via la clé publique(**5**).

Le second protocole, nommé U2F donne la possibilité aux RPs de renforcer la sécurité de leur mot de passe via un facteur d'authentification supplémentaire. Ce dernier pourra être demandé à n'importe quel moment durant la navigation. De plus, il permet de réduire le mot de passe principal à un code PIN classique. Ce principe est basé sur un token physique de type clé USB ou via NFC [13]. De même que son homologue UAF, il possède les deux mêmes phases.

- Lors de l'enregistrement, l'utilisateur active son device via la méthode choisie (Bouton USB, NFC ou autre) et l'enregistre pour son compte au niveau du RP. Le navigateur enverra un hash contenu le nom, le port et le protocole représentant le RP. Sur base de ceci, le device créera une paire de clés. La clé privée sera gardée localement au niveau de ce dernier, tandis que la clé publique sera stockée au niveau du RP et associée à l'utilisateur. Le device enverra par la même occasion une Key Handle avec la requête d'origine encodée dedans. Cette dernière sera utilisée lors de l'authentification.

  Le device possèdera une clé privée pour chaque RP dans lequel il a été enregistré.
- Lors de l'authentification, l'utilisateur entre son mot de passe classique et effectue l'action du second facteur par la suite (USB, NFC etc..). Ceci permettra de vérifier la présence du device. Le RP

13. Near Field Communication permet la communication entre deux appareils électroniques.

va alors envoyer la Key Handle[Sampath Srinivas, 2015] ainsi qu'un hash de la requête d'origine au device via le navigateur. Cette dernière sera utilisée par le device afin d'identifier la clé privée liée au RP et assurer qu'il a bien émis ce hash d'origine. Une fois ceci fait, il créera une signature qui sera renvoyée au RP. Tout ce système permettra d'attester que l'utilisateur possède bien le device. Les deux images présentent en annexe résument tout ceci U2F.

Comme nous pouvons le voir, la sécurité des protocoles FIDO est une priorité. Elle est basée sur la cryptographie à clés publiques. Ceci est réalisable grâce à la standardisation de la couche client servant à l'authentification locale ainsi que celle du protocole de cryptographique utilisé lors de la création des clés. Un point d'attention spécifique a été mis sur l'aspect protection des données privées des utilisateurs. En effet, les données de l'utilisateur enregistrées, comme les données biométriques ou autres seront stockées sur le device et ne transiteront pas via le réseau. De plus, chaque donnée utilisée lors de l'authentification le sera qu'avec l'accord de l'utilisateur. Actuellement, le système U2F ne fonctionne pas encore sur les smartphones.

Même si FIDO est sécurisé, il est intéressant de savoir que ce n'est pas la perle rare qui va résoudre tous les problèmes de sécurité. UAF et U2F possèdent aussi des failles de sécurité. Nikos Leoutsarakos liste quinze problèmes de sécurité[Leoutsarakos, 2016] liés à ces deux protocoles. Il dit notamment que le lien le plus faible reste celui du challenge-response, car peu importe la méthode utilisée par s'authentifier via l'authentificateur, forte ou faible, le niveau de sécurité du protocole de challenge restera le même. Il cite d'autres problèmes, comme des challenges malicieux, certaines attaques MITM sophistiquées ou la date d'expiration des clés infinie.

2.6 Définition et mesure de l'authentification forte

Ce début de chapitre nous a permis de faire un listing des méthodes d'authentification existantes.
Lors de la description de celles-ci, nous avons utilisé un terme qui est revenu très souvent : l'authentification forte.
La sémantique de celui-ci est peut-être un peu vague à la vue du lecteur. Il est donc important d'éclaircir le sujet.
Nous savons que l'authentification par mot de passe est une authentification simple. Elle est d'ailleurs aussi associée à 'l'authentification à un seul facteur' par la plupart des experts en sécurité. Différentes sources comme FIDO Alliance ou Wikipedia[Wikipedia, 2016a] définissent l'authentification forte comme "*au moins la concaténation de plusieurs facteurs d'authentification*". Nous pouvons donc créer un chainage de deux ou plusieurs méthodes afin de renforcer l'authentification générale. Il faut préciser qu'avoir deux fois la même authentification n'est pas très intéressant. Si un attaquant arrive à rendre faillible la première, il arrivera probablement à faire de même pour la seconde. Il est donc plus judicieux que chacune des étapes soit différente.
Cela nous amène à avoir plusieurs cas d'utilisation. Si nous reprenons la définition du cours de sécurité[Colin, 2016]

qui décrit l'authentification comme ce que nous sommes, ce que nous possédons ou ce que nous connaissons alors différents possibilités d'infrastructure ressortent. Pour une sécurité améliorée, il est intéressant d'utiliser au moins deux des trois approches que nous venons de citer.
Partons de l'hypothèse que le premier facteur soit basé sur ce que nous connaissons, comme généralement le mot de passe ou le code PIN, alors nous pouvons classer les prochains facteurs [14] comme suit :

- Sur ce que nous connaissons comme le mot de passe, le code PIN ou encore une phrase secrète. Sachant le premier facteur est similaire, ceci n'apporte pas de grande valeur ajoutée à l'authentification globale.
- Sur ce que nous sommes comme la biométrie. Avoir un second facteur tel qu'une empreinte digitale, vocale ou un scan de l'iris apporte un réel renforcement de la sécurité globale du système.
- Sur ce que nous possédons comme les certificats ou les tokens tels que les clés USB, smartphones, Dongle etc... Incluons aussi les code OTPs vu qu'ils sont souvent associés à un token hardware. Ces solutions apportent aussi un renforcement de la sécurité globale du système.

Les deux derniers facteurs sont les plus répandus, faisant de la biométrie, du certificat, de l'OTP et du token hardware, des moyens permettant l'authentification forte.

Ces différents moyens ont le but de renforcer la sécurité par rapport aux mots de passe. Cependant, il est difficile de savoir lequel est le plus sécurisé de tous. Il aurait été plus facile d'avoir une échelle graduée qui dirait que par exemple le certificat est à 7/10 et la biométrie à 8/10.
Nous pouvons quand même donner une estimation du niveau de sécurité d'une méthode.
Les systèmes basés sur ce que nous sommes sont selon nous les plus sécurisés. La réquisition de la présence physique de l'utilisateur limite la capacité d'attaque d'un hackeur malgré les rejeux d'authentification possible suite au vol de l'empreinte électronique de ce dernier.
Les systèmes basés sur ce que nous possédons ont aussi un bon niveau de sécurité. Cependant, sachant qu'il est plus facile de voler un objet que d'enlever une personne, nous les considérons un peu moins sécurisé que le système expliqué ci-avant. La surface d'attaque reste aussi assez limitée.
Les systèmes basés sur ce que nous connaissons sont les moins sécurisés car le vol de secret est plus simple que les deux autrs. Par conséquent, il peut être soumis à un plus nombre d'attaques.

2.7 Réponse aux questions : Partie I

Dans la première section, nous nous sommes demandés s'il était possible de remplacer totalement les mots de passe afin de pallier au problème de sécurité qui les concerne ou à défaut de les remplacer, proposer des alternatives.
Grâce au listing établi, nous avons remarqué que le remplacement définitif était possible mais restait très difficile. Les descriptions ci-dessus démontrent qu'il existe des solutions potentielles qui proposent un compromis correct entre coût, utilisabilité et sécurité.

14. Pour notre classement, nous nous limiterons à deux facteurs

Les méthodes basées sur l'authentirfication forte et les proxies pourraient être de très bons candidats au remplacement définitif des mots de passe. Cependant, ils existent depuis plusieurs années, et même s'ils sont assez répandus sur le Web, nous ne voyons pas de tendance à une utilisation omniprésente de ceux-ci.

Actuellement, seuls les sociétés et les utilisateurs aguéris implémentent ce type d'authentification. Ceci est expliquable pour diverses raisons. La première est le fait que certaines de ces solutions sont coûteuses, donc limitant l'accès à une partie de la communauté des utilisateurs. La seconde raison est l'utilisabilité du point de vue des utilisateurs. Les systèmes d'authentification peu habituel, ou trop complexe nécessitent un apprentissage des utilisateurs, ce qui est un frein à leur adoption.

CHAPITRE 3

L'authentification unique

Lors du chapitre précédent, nous avons classé les différentes méthodes d'authentification en distinguant les moyens d'authentification et les protocoles. Nous allons maintenant nous intéresser à un protocole en particulier, le SSO. Le domaine du SSO est vaste et comporte beaucoup de méthodes et de protocoles différents. Une étude plus précise de celui-ci apportera une vision plus claire sur le sujet pour le lecteur. Dans un premier temps, nous allons le décrire en distinguant les protocoles des implémentations. La plupart des implémentations de SSO existantes se basent sur les principes de protocoles SSO et y apportent des modules ou fonctionnalités supplémentaires. Nous verrons que les protocoles SSO appartiennent uniquement à la famille des True-SSO.
Ensuite, nous ferons une comparaison entre True et Pseudo SSO sur base de différents critères. Nous verrons que chacun d'entre eux impliquent une infrastruture spécifique. Pour ce faire, nous réaliserons des tableaux comparatifs entre ces systèmes.

Si nous revenons au sujet de l'authentification en général et à la problématique des mots de passe évoquée lors les chapitres précédents, il serait intéressant de se demander si un système de SSO global pourrait voir le jour.
Serait-il possible un jour d'avoir un SSO sécurisé, utilisable et peu coûteux permettant d'accéder à n'importe quel site présent sur le Web ?
Nous tenterons d'apporter un élément de réponse à cette question à la fin de ce chapitre.

3.1 Les protocoles

Les protocoles SSO ont été développés en suivant des normes et des spécifications bien particulières. Nous dintinguons quatre protocoles différents : CAS, SAML 2.0, OAuth 2.0 et OpenID Connect.

3.1.1 CAS

CAS est un système de SSO orienté Web développé en JAVA. Son principe est similaire à celui des tickets Kerberos[Mathieu, 2004]. Il possède quatre types de tickets différents, pour deux modes de

fonctionnement distincts[Pascal Aubry, 2004] : TGT, ST, PGT et PT. Ces tickets ne contiennent aucune information. Ils sont dits opaques. Les deux finalités sont le mode non-proxy et le mode proxy.

- Le mode non-proxy utilise les tickets TGT et ST. A la première connexion au serveur CAS, l'utilisateur devra s'authentifier via un couple d'identifiants classique. Si cette étape est réussie, le serveur CAS enverra un TGT au navigateur, qu'il stockera dans ses cookies. Ce dernier possède une durée de vie limitée, équivalente à quelques heures. Son rôle est de permettre au navigateur de recevoir des ST du serveur CAS pour les applications clientes. Lors d'un accès à une zone protégée d'une application CASifiée[1], le navigateur sera redirigé vers le serveur CAS et lui fournira son TGT. Voyant que le navigateur possède le TGT, le serveur créera un ST à usage unique et lui transmettra. Ce ST sera lié à l'application source et possèdera une durée de vie très courte. Le navigateur sera alors redirigé vers l'application qui validera le ST directement avec le serveur CAS. La validation se fera en HTTP. Celle-ci faite, la zone protégée sera alors affichée dans le navigateur. L'ensemble de ces étapes est représentée dans le schéma ci-dessous :

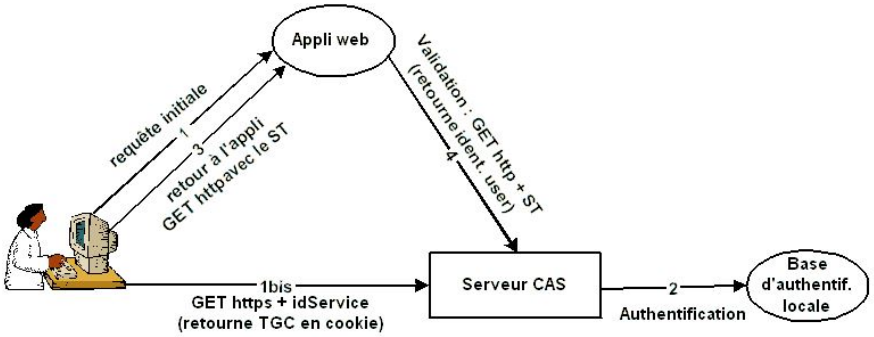

FIGURE 3.1 – CAS en mode non-proxy

- Le mode proxy est un peu différent de ce que nous venons de voir. Il utilise des PGT et PT. Afin de pouvoir délivrer des tickets dans ce mode de fonctionnement, le serveur CAS va valider que le navigateur possède un PGT. Celui-ci est l'équivalent du TGT dans le mode non-proxy. Il permet à une application agissant en tant que proxy d'authentifier l'utilisateur auprès du serveur CAS et de recevoir des PT (tout comme le PGT, le PT est l'équivalent du ST) pour les autres applications non-proxy. Quand une application reçoit son PT, la validation est similaire à celle présentée ci-dessus pour les ST. L'idée derrière cette architecture est d'avoir un seul point d'entrée pour l'utilisateur : l'application proxy. C'est elle qui demandera les PT pour les autres applications derrière ce proxy.

1. Jargon classique qui veut dire que l'application est impliquée dans le SSO CAS.

Le schéma ci-dessous illustre ce principe :

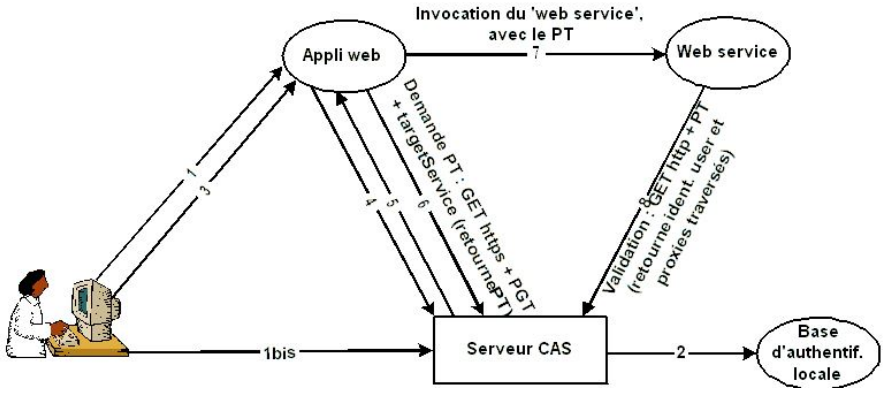

FIGURE 3.2 – CAS en mode proxy

Ces deux modes ont des buts différents. Le mode non-proxy est destiné à un SSO entre applications, sans avoir de point d'entrée spécifique. Par contre, le mode proxy fonctionne souvent avec des portails de liens. L'utilisateur voit généralement une seule application, le portail lui-même. Les redirections vers les autres applications sont transparentes et il ne se doute pas qu'il puisse y avoir plusieurs applications derrière ce même portail. Ce cas d'utilisation est souvent utilisé par les petites entreprises, écoles ou Webmail.

CAS possède trois avantages d'un point de vue sécurité.

- Les secrets ne passent que du navigateur vers le serveur d'authentification de manière chiffrée.
- Se re-authentifier se fait de façon transparente tant que le navigateur possède un TGT/PGT. De plus, comme déjà dit, ce ticket est opaque et n'est présenté qu'au serveur d'authentification en HTTPS.
- Les applications connaissent l'identité des utilisateurs uniquement via les tickets ST/PT. De plus, ces derniers sont temporaires et sont toujours validés au préalable par le serveur d'authentification lors de l'accès.

Mais il possède aussi des contraintes et désavantages. Le maillon faible de la chaine est le serveur CAS lui-même. S'il est compromis ou inaccessible, alors tout le système est en péril. Il y a une contrainte majeure, si le navigateur de l'utilisateur n'accepte pas les cookies, alors le SSO devient complètement inutile. De plus, les applications Web utilisant CAS doivent implémenter une librairie spécifique.

CAS reste aussi limité par le fait que la seule méthode d'authentification possible sur le serveur CAS est un couple d'identifiants classique. Ce concept ne supporte pas d'autres méthodes telles que l'authentification par certificat, OTP ou encore multi-facteurs. Et enfin, il ne permet pas de faire du MDSSO.

3.1.2 OAuth 2.0

OAuth 2.0 est un protocole Web qui permet de faire du SSO via un système de délégation de droits entre SPs et IdPs. Les droits donnés ne peuvent se faire qu'avec l'accord de l'utilisateur final.

Pour que le SSO se fasse avec OAuth, un site Web doit jouer le rôle d'IdP. L'utilisateur connecté, les SPs annexes viendront vérifier l'identité de l'utilisateur auprès de l'IdP.

Les échanges entre applications se font via des tokens. Il en existe plusieurs :

- Les access tokens sont les plus importants puisqu'ils donnent les droits au SP.
- Les refresh tokens interviennent lorsque l'access token est expiré ou invalide. Il servira à redemander un nouveau access token.
- Les request tokens interviennent lors de la demande d'accès pour le SP.

Pour faire en sorte qu'une application utilise OAuth, il faut l'enregistrer. Afin de réaliser cette étape, des informations, telles que le nom de l'application, le site Web ou encore le logo vont être demandées. Il sera aussi important d'indiquer l'URI qui servira lors des redirections des utilisateurs. Une fois l'application enregistrée, le client recevra alors un ID et un secret. Cet ID est exposé publiquement et est utilisé par les API OAuth pour identifier l'application. Le secret, quant à lui, est utilisé pour authentifier l'application auprès de l'API quand l'application demande l'accès aux données via un request token.
OAuth définit quatre rôles comme le décrit ce blog[Reinke, 2013] :

- Le Resource Owner, représente l'entité possèdant les données. Il peut donner accès aux données.
- Le Resource Server représente le serveur qui conserve les données qui sont protégées.
- Le Client Application représente l'entité demandant accès aux données protégées.
- L'Authorization Server fournit les tokens nécessaires pour pouvoir accéder (temporairement) aux données.

Sun[Sun and Beznosov, 2012] distingue deux flux d'authorisation différents :

- Le Server Flow, plus connu sous le nom de "Authorization Grant Code". Ce flow intervient lorsqu'un SP reçoit un access token de l'application tournant du côté serveur, et donc où le code n'est pas public.
- Le Client Flow, plus connu sous le nom de "Implicit Grant". Il est utilisé lorsqu'un SP reçoit un access token via un Javascript tournant dans le navigateur. Par rapport au flow précédent, celui-ci est destiné aux applications qui ne peuvent embarquer un secret comme les applications Javascript.

Le schéma, tiré de la documentation officielle oAuth[D. Hardt, 2012] ci-dessous représente l'acquisition d'un token d'accès par la machine cliente, afin d'accéder à une ressource protégée.

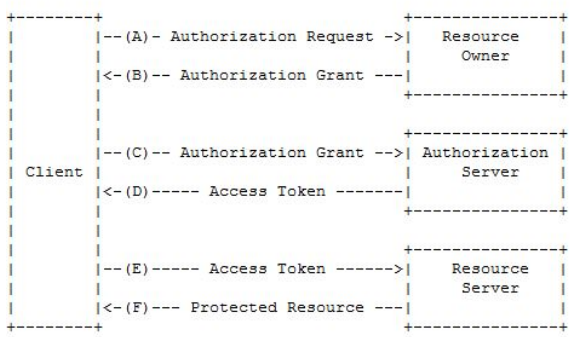

FIGURE 3.3 – Demande d'un token d'accès

Nous pouvons distinguer plusieurs types de clients OAuth, ce qui implique plusieurs profils possibles. Voici les deux types de clients :

- Le premier type est le type confidentiel. Cela veut dire que l'application peut garder le secret du client de manière sécurisée.
- Le deuxième type ne permet pas de garder le secret du client de façon sécurisée. Ceci est typiquement le cas d'une application mobile ou desktop, qui garde son secret de façon embarquée.

Voici les différents types de profils existants :

- Le profil "Web Application" représente une application tournant côté serveur sur lequel le secret est stocké sur le serveur. Le type est donc confidentiel.
- Le profil "User Agent Application" représente typiquement les applications Javascript. Ce type est donc public.
- Le profil "Native Application" représente les applications mobile ou desktop. Ce type est public.
- Le profil "Hybrid Applications" permet de mixer les trois profils précédents. OAuth ne définit pas de quel type il s'agit dans sa documentation.

L'utilisation de OAuth est principalemnt liée aux applications sur Internet afin de pouvoir faire du MDSSO de façon rapide. Cependant, il peut très bien être utilisé en interne lors d'un SSO classique. Mais ceci reste quand même assez rare.
A titre d'exemple, beaucoup d'applications de jeu utilisent oAuth avec Facebook ou encore Twitter. Les réseaux sociaux sont friands de ce mécanisme, leur apportant de nouveaux utilisateurs et renforçant leur présence sur le Web.
Le schéma ci-dessous est exemple d'interraction avec Twitter et une application tierce :

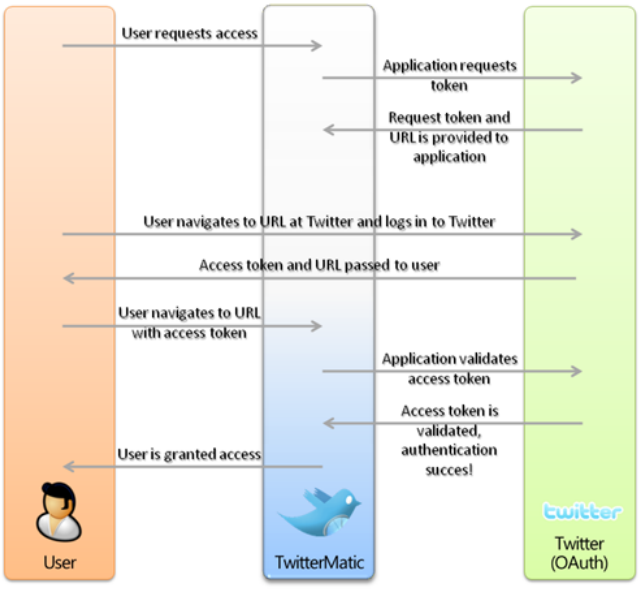

FIGURE 3.4 – oAuth avec Twitter [Balliauw, 2009]

Selon Sun[Sun and Beznosov, 2012], OAuth comporte plusieurs désavantages. Premièrement, OAuth est un protocole qui n'utilise aucune protection cryptographique comme la plupart des protocoles de sécurité. Cela induit que les SPs doivent utiliser SSL pour contourner ce problème, ce qui est peu fréquent [2] . Deuxièmement, dans le cas d'un Client Flow, les access tokens passent par le navigateur, ce qui augmente les possibilités d'attaques telles que XSS, CSRF et l'usurpation d'identité. En effet, la clé secrète de l'utilisateur peut être compromise en copiant ou en créant un access token qui aurait les mêmes propriétés de portée et de temps d'accès aux ressources que l'original. Un attaquant en possession de celui-ci pourrait alors obtenir les mêmes droits que l'utilisateur. Même si OAuth est considéré comme sécurisé de par ses spécifications, ce type d'attaque est réalisable par le fait que les SPs sont mal sécurisés ou via des failles dans le navigateur.

3.1.3 OpenID Connect

OpenID Connect[Pashalidis and Mitchell, 2003] (OIDC) est un protocole d'authentification basé sur les mêmes spécifications que OAuth 2.0 . Il représente l'évolution d'OpenID 2.0, qui était son prédécesseur. Il permet un SSO entre applications Javascript basées sur un navigateur, applications Web ou applications mobiles natives[Nat Sakimura, 2014]. Son principe est d'autoriser les applications à venir vérifier l'identité de l'utilisateur auprès de l'IdP. Cette vérification est effectuée via des tokens JWT échangés au travers une interface REST. Ce token contient le résultat de l'authentification, la date d'expiration, l'émetteur, l'identifiant du sujet ainsi que toute autre information utile lors du flux d'authentification.
OIDC fonctionne en deux phases : l'enregistrement et l'authentification.

- Durant l'enregistrement, le SP va devoir envoyer un message POST au Client Registration Endpoint [3] avec les Metadata le concernant. Ces dernières serviront à l'identifier. L'IdP va alors assigner un identifiant unique (ID) à ce dernier et associera les Metadata envoyées dans le message à ce même ID. Il liera aussi une méthode d'authentification spécifique au SP. Il en existe cinq :
  1. Le client_secret_basic : il permet l'authentification basique via HTTP.
  2. Le client_secret_post : les identifiants de l'utilisateur sont incorporés dans le corps de la requête.
  3. Le client_secret_jwt : l'authentification se fait via le token JWT , précédemment créé.
  4. La private_key_jwt : le SP reçoit une clé publique avec laquelle il va signer le JWT créé. Ce dernier sera utilisé lors de l'authentification. Cela permet une authentification plus sécurisée.
  5. None : le SP ne s'authentifie pas auprès du service OIDC , soit parce qu'il est public ou parce qu'il utilise l'Implicit Flow ou encore parce qu'il utilise un autre mécanisme d'authentification.

  Si la phase d'enregistrement est réussie, l'IdP renverra au SP son ID, une URL lui étant associée ainsi que le secret, si créé auparavant. Tout ceci se passe via des messages JSON, nécessitant donc l'utilisation cette technologie au niveau du code des services clients.

2. L'article de Sun dit "que seulement 21% des SPs qu'il a évalués utilisent SSL pour protéger les sessions SSO".
3. Composant d'enregistrement.

- L'authentification peut être établie soit de l'IdP, soit du SP. Dans le premier cas, l'IdP redirigera l'utilisateur vers le SP qui devra effectuer une requête d'authentification auprès de celui-ci. Dans le second cas, aucune redirection n'est nécessaire. Lors de l'envoi de la requête d'authentification du SP vers l'IdP, celui-ci transmettra l'ID de l'utilisateur ainsi que l'URL associée.
  L'authentification peut suivre trois chemins différents : l'Authorization Code Flow, l'Implicit Flow ou l'Hybrid Flow. Le tableau de la documentation officielle résume leurs fonctionnalités principales et se trouve dans l'annexe Authorization Code Flow.
  L'Authorization Code Flow va permettre de retourner un code d'autorisation au SP, qui pourra alors l'échanger contre un token ID ou un token d'accès. Ceci se fait à travers huit étapes. Ceci est disponible dans la même annexe que ci-dessus Authorization Code Flow.
  L'Implicit Flow est utilisé dans le cas d'une application de type Javascript dans le navigateur. Aucune authentification n'est faite par l'IdP. Le token d'accès et le token ID sont directement renvoyés à l'application qui peut les exposer à l'utilisateur ou à toute autre application disposant d'un accès au navigateur de l'utilisateur. Ceci comporte bien sur un certain risque de vol de token et d'usurpation d'identité. De la même façon que pour le flow précédent, ceci est réalisé via plusieurs étapes décrites en annexe Implicit Flow.
  L'Hybrid Flow fonctionne différemment. Lorsque ce dernier est utilisé, il retournera des tokens de l'Authorisation EndPoint [4] et des tokens du Token EndPoint [5]. Ce mécanisme est spécifique à OAuth 2.0. Les étapes de ce flow sont décrites en annexe Hybrid Flow.

OIDC permet la création de profil et donc une certaine gestion de droits. Ceci est réalisé via des attributs utilisateurs, appelés Claims ainsi qu'un attribut scope. Ceux-ci sont récupérés lorsque le SP fait une requête avec son token d'accès au composant Userinfo Endpoint [6]. Ce dernier retournera alors un objet JSON dans lequel les attributs seront présents. Lors d'un flux de ce type, au minimum deux attributs sont envoyés : l'émetteur et le sujet. Ils permettent de réaliser l'unicité d'un utilisateur. Ces attributs sont transférés du serveur d'autorisation via le JWT.

Le SSO d'OIDC comporte plusieurs avantages du point de vue de la sécurité. En effet, ses mécanismes de signature et d'encryption (JWS + JWE) vont permettre d'assurer l'intégrité et la confidentialité des échanges. L'intégrité est respectée par la signature des messages JSON via JWS. La confidentialité est quant à elle conservée via l'encryption JWE de ceux-ci. De plus, l'ensemble des communications entre SP et IdP se fait via le protocole SSL ou TLS, impliquant donc la gestion de PKI et la mise en place d'une relation de confiance entre eux. De plus, dans le but d'éviter toute corrélation des utilisateurs au sein des SPs, un mécanisme de pseudonyme est possible.
La notion de protection des données personnelles n'est pas ignorée non plus. L'accord de l'utilisateur est obligatoire lors du transfert d'informations. De même, au niveau des SPs, seul le minimum d'informations nécessaires sera stocké.

4. Ce dernier effectue l'authentification de l'utilisateur.
5. Composant duquel sont envoyés les tokens ID, d'accès et de refresh
6. Composant qui possède les informations relatives à l'utilisateur.

Il existe certains désavantages avec OIDC. En premier lieu, comme dans la plupart des SSO, si le compte principal est compromis, alors l'attaquant aura accès à l'ensemble des SPs avec les droits ou profil(s) de l'utilisateur. Ceci est généralement réalisé avec des attaques de type phishing[Sun et al., 2010]. En second lieu, l'utilisabilité de l'interface des SPs est parfois un frein à l'adoption d'OIDC. Les utilisateurs sont obligés de s'accoutumer des interfaces de chaque SP. En dernier lieu, OIDC permet l'authentification avec plusieurs IdPs. Tout utilisateur peut implémenter son propre serveur OIDC, impliquant donc la possible existance de serveurs mal sécurisés ou malicieux.

3.1.4 SAML 2.0

SAML 2.0 est un protocole dont le but est de faire des échanges de messages sous format XML[7] de façon sécurisée entre domaines Web. La documentation officielle[OASIS-Open, 2006] donne une vue architecturale de haut niveau des composants qui permettent ces échanges :

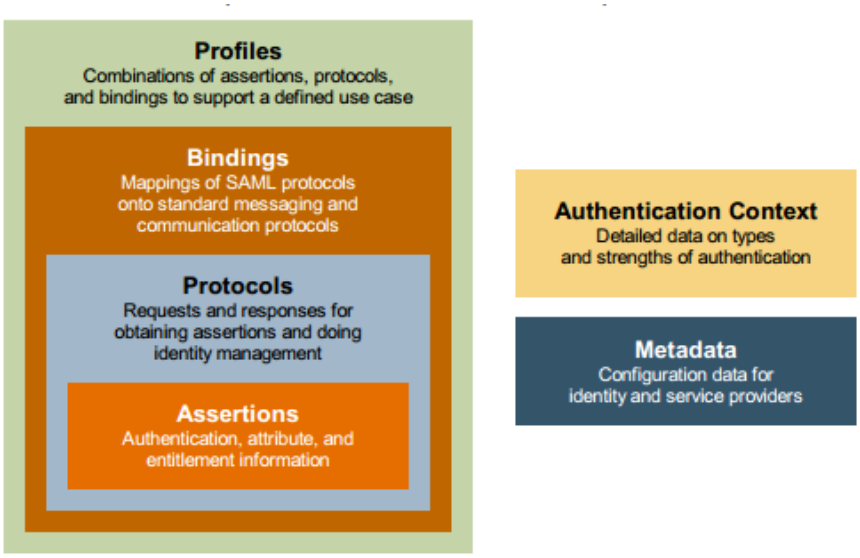

FIGURE 3.5 – Interactions entre composants basics

Sur base de cette dernière, nous allons décrire les différents composants présents sur l'image ci-dessus :
L'assertion SAML contient trois types d'information :

- Des informations concernant l'authentification tels que le statut et l'heure à laquelle elle a été réalisée par l'utilisateur.
- Des informations concernant les autortisations. Cela comprend les permissions liées à l'utilisateur.
- Des informations concernant les attributs utilisateurs.

SAML définit six sortes de protocole :

- L'Authentication Request Protocol est un protocole qui définit un moyen avec lequel une entité peut demander des assertions contenant des informations à propos de l'authentification. Cette requête peut aussi demander des attributs avec celle-ci. Ce protocole est principalement utilisé par

7. On les appelle aussi assertions.

le profil Web Browser SSO dans le cadre d'une redirection d'un SP vers l'IdP afin de recevoir les informations de sécurité adéquates.

- Le Single Logout Protocol permet d'effectuer un logout ou fermeture de session d'un utilisateur de manière quasi instantanée.
- L'Assertion Query and Request Protocol permet de définir un ensemble de requêtes avec lesquelles les assertions peuvent être obtenues.
- L'Artefact Resolution Protocol permet de passer des messages par référence en utilisant une valeur d'une longueur fixée. Cette valeur est appelée un artefact. L'entité recevant l'artefact va utiliser le protocole afin de faire l'opération inverse et recevoir le message d'origine. L'artefact est généralement passé en utilisant les bindings.
- Le Name Identifier Management Protocol permet de changer la valeur ou le format d'ID utilisé pour référencer l'entité.
- Le Name Identifier Mapping Protocol permet de faire un mapping entre un ID (name identifier) en un autre ID.

Les Bindings SAML permettent de détailler de manière précise l'envoi des messages. Il en existe plusieurs :

- Les HTTP redirect Binding sont utilisés dans le cadre d'une redirection 302.
- Les HTTP POST Binding définissent comment les messages sont envoyés dans les formulaires encodés en base 64.
- Les HTTP Artifact Binding définissent comment les artefacts sont transportés d'un envoyeur vers un receveur. Cela se fait soit via un formulaire HTML ou via une query String dans l'URL.
- Les SAML SOAP Binding définissent comment les messages sont envoyés à travers les échanges SOAP.
- Les Reverse SOAP Binding définissent les échanges d'un flux permettant à un client HTTP d'être un répondeur SOAP. Ils sont utilisés dans le cadre de client "Enhanced Client", de "Proxy Profile" et particulèrement dans celui des passerelles WAP.
- Les SAML URI Binding définissent comment récupérer une assertion SAML d'une URI.

Les composants précédents permettent de former les profils. Ils représentent les cas d'utilisation entre SPs et IdPs :

- Les Web Browser SSO Profile définissent comment les entités SAML utilisent les Authentication Request Protocol et SAML Response messages ainsi que les assertions afin de pouvoir mettre en place le SSO avec les navigateurs. Cela implique la définission des combinaisons entre HTTP Redirect, POST et Artifact Bindings. C'est le profil qui nous intéresse le plus dans le cadre de ce chapitre.
- Les Enhanced Client and Proxy Profile (ECP) définissent un profil SSO spécial destiné aux clients et passerelles proxy qui utilisent les Reserves SOAP et SOAP bindings.
- Les Identity Provider Discovery Profil définissent un mécanisme permettant au SP de découvrir les IdP précédemment visités.

- Les Single Logout Profile définissent comment les protocoles Single Logout Procol peuvent être utilisés avec SOAP , HTTP Redirect, HTTP POST et HTTP Artifact Bindings.
- Les Assertion Query / Request Profile définissent comment les entités peuvent utiliser les Query et Response Protocol pour obtenir les assertions SAML lors d'un binding synchrone, comme par exemple SOAP.
- Les Artifact Resolution Profile définissent comment les entités peuvent utiliser les Artifact Resolution Protocol pour obtenir les messages référencés par un artefact à travers un binding synchrone.
- Les Name Identifier Management Profile définissent comment peuvent être utilisés les Name Identifier Management Protocol avec SOAP, HTTP POST, HTTP Redirect et HTTP Artifact Bindings .
- Les Name Identifier Mapping Profile définissent comment les Name Identifier Mapping Protocol peuvent être utilisés avec des bindings synchrones comme SOAP.

Tous ces composants permettant deux cas d'utilisation généraux : le Web Browser SSO et la fédération d'identité.

Le premier cas permet de faire du SSO via le navigateur et est donc dédié aux applications Web. La persistence de la session est conservée via un mécanisme de cookie. La particuliarité de ce SSO est qu'il permet l'authentification unique entre domaines Web tout en ayant une sécurité renforcée. Un flux avec SAML fait intervenir les acteurs classiques : IdP et les SPs avec une relation de confiance préalablement établie entre eux. Le principe sera alors de transférer une assertion de l'IdP vers le SP [8]. Le SP l'analysera et donnera l'autorisation à l'utilisateur. Le schéma ci-dessous donne une vue détaillée d'un échange SSO[OASIS-Open, 2006] :

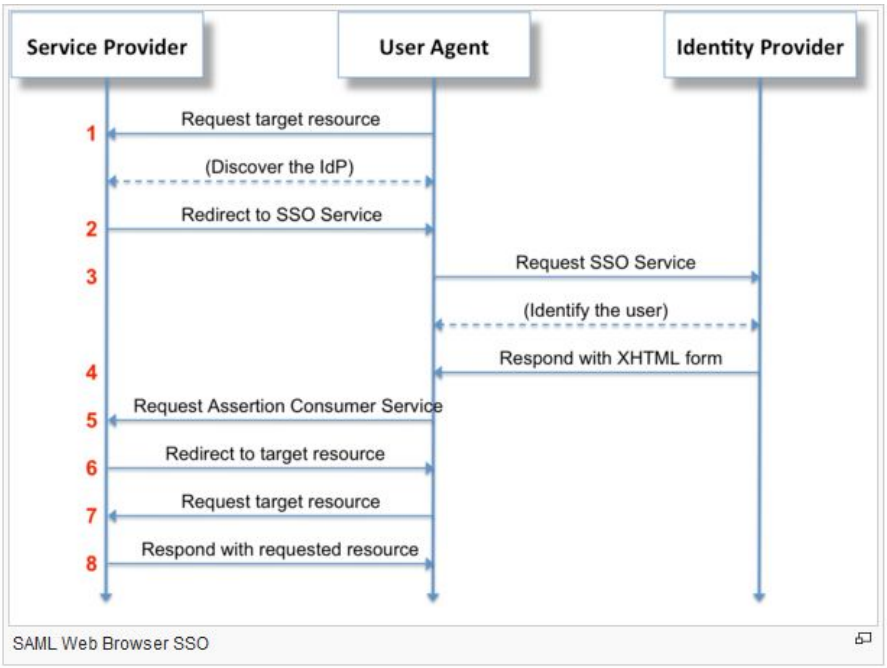

FIGURE 3.6 – SAML Web Browser SSO

8. Ceci implique une connexion sécurisée de type SSL / TLS

En (**1**), l'utilisateur va envoyer une requête afin d'accéder à une ressource protégée, présente sur le SP. Celui-ci va effectuer une phase de découverte ou 'discover' afin de détecter l'IdP et rediriger le navigateur vers ce dernier , qui va identifier l'utilisateur et lui présenter un formulaire . Cette phase représente les numéros (**2**, **3** et **4**) sur le schéma. Les phases (**5**,**6**,**7**,**8**) vont servir à récupérer les assertions de l'IdP afin de présenter la ressource protégée au navigateur. SAML dans ce mode de fonctionnement est souvent associé à la fédération d'identité, afin de permettre une identité unique au travers l'ensemble des domaines. Cette dernière est expliquée en annexe Fédération d'identité.

SAML est particulièrement utilisé au sein de grandes entreprises. Ceci est au du au fait qu'il comporte un grand nombre d'avantages.

1. Il est totalement indépendant de la technologie ou de la plateforme utilisé. SAML est fortement maniable et peut s'intégrer avec tout type de plateforme, permettant donc une interopabilité entre systèmes[Lewis and Lewis, 2009].
2. Il permet de faire du MDSSO.
3. Il permet la fédération d'identité lors du MDSSO. Cela permet de créer des partenariats entre différents domaines et donc entre entreprises.
4. Il permet une gestion des attributs et des autorisations pointues.
5. Il permet le chiffrement et la signature XML.
6. Le SSO est simple à utiliser pour l'utilisateur final.

Cependant, SAML est quand même assez complexe et ne permet l'ajout de SPs dynamiquement dans sa fédération.
Le principe SAML est la base du développement de certains frameworks comme WS-Security, XACML ou encore STORK / CEF eID.

3.2 Les implémentations

Nous avons expliqué les protocoles ci-dessus. Les implémentations sont entièrement basées sur ceux-ci. Les protocoles sont souvent considérés comme incomplets en terme de fonctionnalités. De ce fait, certaines sociétés ont développés des solutions plus complètes se basant sur leur principe mais en y apportant les modules supplémentaires.

3.2.1 Facebook Connect

Facebook Connect, aussi appelé Facebook Login, est un protocole propriétaire qui est utilisé principalement par des applications tierces pour authentifier les utilisateurs. Selon Serge Egelman[Egelman, 2013], il y aurait huit millions de sites et applications qui s'en servent.
Facebook Connect possède deux couches :

- Une couche permettant d'avoir de gérer les accès aux informations de manière sécurisée. Cette couche est entièrement basée sur OAuth 2.0. Elle va permettre aux applications tierces d'accéder aux informations du profil d'un utilisateur. Sur base de celles-ci, certains accès seront établis. Il est intéressant de noter que Facebook va plus loin et permet aux applications d'avoir un rôle en lecture mais aussi un rôle en écriture. Il joue donc plus que le rôle de vérificateur d'identité d'OAuth 2.0. Les applications peuvent donc mettre à jour certaines données du profil de l'utilisateur Facebook. Typiquement, il a été remarqué que des applications publient des liens, des photos, des commentaires ou encore des vidéos sur le mur des utilisateurs. Les droits et autorisations en lecture qui sont accordés aux applications sont gérables via l'accord de l'utilisateur. Il peut ainsi avoir le contrôle sur les données transférées.
- L'autre couche permet l'authentification et a été développée par Facebook. C'est une authentification classique avec le couple d'identifiants tels que nous les connaissons. Facebook a voulu ajouter cette couche pour répondre au fait que les fonctionnalités que propose OAuth 2.0 sont limitées[9].

Voici un schéma d'une application demandant accès aux données d'un utilisateur Facebook.

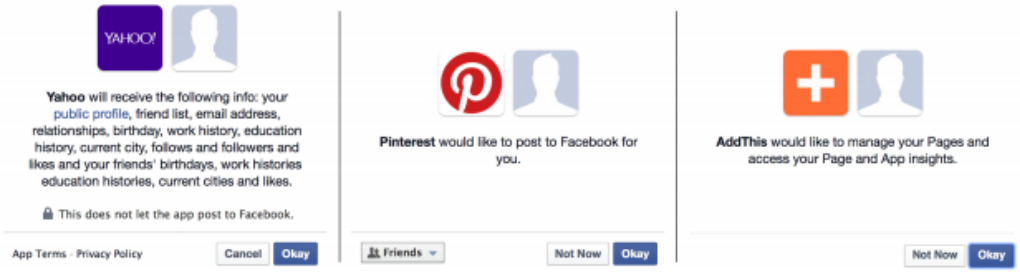

FIGURE 3.7 – Demande d'accès d'une application via Facebook Connect

Quand un utilisateur veut ajouter une application tierce à son profil Facebook, une boite de dialogue va lui être affichée, lui demandant son consentement[Wang et al., 2011]. S'il accepte, un ensemble d'informations l'identifiant va transiter vers l'application cible. Néanmoins, Facebook permet de désactiver les informations envoyées aux applications tierces. Techniquement, selon Robinson et Bonneau[Robinson and Bonneau, 2014], trois types de requêtes existent :

- Les requested permissions, qui sont des requêtes de demande permission faite par l'application via l'API de Facebook Login.
- Les granted permissions, qui sont la 'traduction' des requested permissions afin de pouvoir les approuver par l'utilisateur.
- Les displayed permissions, qui sont l'affichage des granted permissions dans une boite de dialogue affichée à l'utilisateur.

Les sessions sont gérées sur base des cookies, comme beaucoup d'autres SSO le font.
Les utilisateurs autorisant l'accès à certaines applications ne se rendent pas toujours compte qu'elles auront aussi accès à des données privées et qu'elles pourraient les diffuser publiquement.

9. Limitées par le fait que OAuth ne fait que de la gestion d'autorisation

D'un point de sécurité, plusieurs études ont démontré que la conservation et l'utilisation des données privées de l'utilisateur restaient très nébuleuses. De plus, les utilisateurs ne sont pas toujours conscients de comment tout ce système de collecte de données est utilisé.
Facebook Connect étant basé sur OAuth 2.0, toute faille de sécurité liée à ce protocole l'affectera aussi.

3.2.2 Cloud-based SSO

Les Cloud-based SSO sont destinés aux applications Web se trouvant dans le Cloud. Ils proposent une liste de liens vers ces d'applications et permettent de naviguer entre elles de façon transparente, sans s'authentifier. Les SSO comme OneLogin[Yahoo, 2016a] de Yahoo ou GoogleApps[Google, 2016a] jouent le rôle d'IdP au sein du Cloud. Ils sont basés sur le protocole SAML et utilisent des certificats X.509 afin d'établir une relation de confiance avec les SPs. Généralement les Cloud-based SSO permettent de gérer les SPs via deux méthodes. Soit l'administrateur du SSO va se connecter via un couple identifiant et mot de passe à la console d'administration, soit via une API qui est fournie par la plupart des SSO. L'API est plus flexible puisqu'elle permet une gestion des autorisations, basée sur l'identité de l'utilisateur. Certains grands acteurs du Web comme Google[Google, 2016b], Microsoft avec son SSO Live Connect[Microsoft, 2016] ou Yahoo[Yahoo, 2016b] mettent à disposition des API OAuth ou OIDC et jouent le rôle d'IdP. Concrètement, cela permet aux utilisateurs de lier certaines applications avec leur compte email hébergé chez eux.

De manière générale, les Cloud-based SSO sont plus sensibles à une compromision ou une indisponibilité de l'IdP puisque si le module est compromis, il pourrait affecter l'ensemble des clients qui utilisent cet outil en ligne. Cela entraine donc qu'ils soient une cible de choix pour les attaquants. Si un d'entre eux accèdait par exemple à la console d'administration du système, il pourrait alors faire ce qu'il veut avec l'ensemble des profils utilisateurs.

3.2.3 ECAS

ECAS est un système de SSO développé par la Commission Européenne. Son but est de permettre aux utilisateurs de voyager à travers l'ensemble des sites mis à disposition en fonction de leur propre pays. Ce système est basé sur CAS mais il est plus complet car il permet de gérer les identités via des droits bien spécifiques.
ECAS est subdivisé en deux parties distinctes[HÄRTWICH, 2014] :

- Une partie SSO classique, basée sur les cookies du navigateur.
- Une partie gestion d'identité via IAM. C'est un système permettant de faire de la gestion d'identités et d'accès. IAM utilise la notion de rôle afin de pouvoir créer l'ensemble des profils utilisateurs et d'y apporter une granularité très fine. De ce fait, la sécurité en est renforcée.

Pour pouvoir utiliser ce système, l'utilisateur doit s'enregistrer en utilisant l'adresse email de son organisation au sein de la Communission Européenne. Le système associera un compte ECAS à cet email. La documentation du projet[HÄRTWICH, 2014] parle de 1 personne = 1 email = 1 compte ECAS. Ce

compte fait office d'identifiant unique et est lié à un numéro unique, appelé PIC number, représentant l'organisation à laquelle il appartient. Il sera aussi lié à différents rôles, représentant les projets dans lesquels il est impliqués.

D'un point de vue sécurité, tout ce système permet de bien délimiter l'espace de chaque personne et donc d'éviter toute dérive dû à des droits trop permissifs. A la commission, la gestion des accès est très importante puisque n'importe qui ne peut accéder à certaines zones privées ou documents spécifiques.
ECAS permet la signature électronique de documents. Il garantit la non-répudiation puisque les clés générées lors de la signature sont liées au compte de l'utilisateur connecté et donc implicitement à son email.
ECAS peut-être utilisé avec le framework CEF eID. Cela permet aux utilisateurs de ne plus être restreints aux services de leur propre pays mais d'avoir un accès à ceux des autres pays également.

3.2.4 STORK 2.0

STORK est un projet de recherche financé par la Commission Européenne dont le but est de pouvoir s'authentifier avec la carte d'identité eID à travers les frontières des pays. Actuellement, il est possible de se déplacer physiquement dans un autre pays de l'UE et présenter sa carte d'identité comme document officiel. Cependant d'un point de vue électronique , cela n'est pas le cas [10]. L'utilisation de la carte d'identité est limitée aux services au sein d'un même pays.
STORK 2.0 est le successeur de STORK 1.0, qui était en phase de test. C'est un framework complet et fonctionnel. Par rapport à la version 1.0 qui était principalement axée sur l'authentification des citoyens et les autorisations qui y sont liées, la version 2.0 introduit de nouveaux attributs , ce qui donne une approche plus flexible. STORK 1.0 ne permettait pas de représenter une entité légale ou encore de pouvoir représenter une autre personne, ce que STORK 2.0 permet. A titre d'exemple, un parent pourrait représenter son enfant auprès d'un service lié à la santé.
STORK 2.0 a été implémenté pour quatre secteurs pilotes : eBanking, eLearning et qualifications académiques, services publics et enfin le secteur de la santé. Le cadre dans lequel ce projet est actif ne concerne que les pays faisant partie de l'UE et qui ont implémenté la technologie de la carte d'identité électronique. Le framework permet une interopérabilité entre les différents services gouvernementaux des pays impliqués. Cela veut dire qu'un utilisateur pourrait s'authentifier avec sa carte d'identité auprès d'un service d'un autre pays.
STORK 2.0 définit deux modèles[Herbert Leitold, 2015] :

- Le premier modèle est basé sur une passerelle nationale par laquelle toutes les transactions vont se faire. Chaque pays en possède une. Cette dernière agit en tant que proxy, cachant l'infrastructure et les spécificités des tokens eID des utilisateurs des autres pays membres impliqués dans les transactions. Ce modèle est appelé 'Proxy-Model' ou encore 'Centralized Deployment Model'.

10. Ce n'était pas le cas au moment de la création du projet

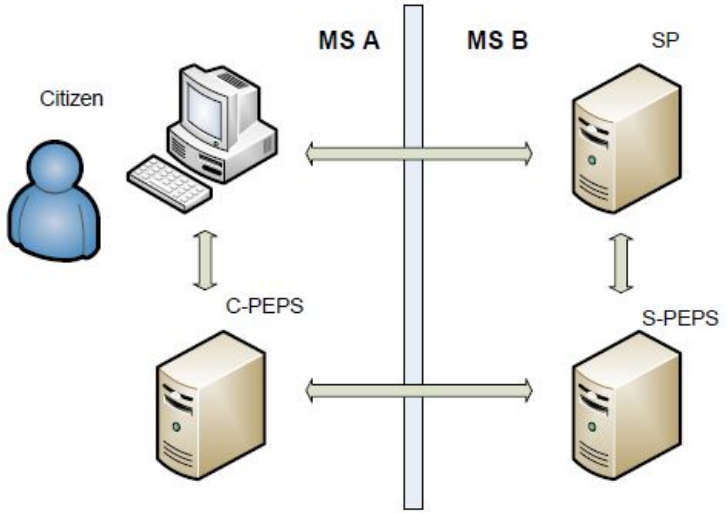

FIGURE 3.8 – Le modèle Proxy

Dans la figure présentée ci-dessous, le MS A accède à un SP appartenant au MS B. Le SP va alors déléguer l'authentification à sa passerelle nationale S-PEPS. Cette dernière va alors transformer le protocole du MS B en un protocole commun qui lui est connu, le protocole STORK. La passerelle va ensuite renvoyer la requête d'authentification vers le module C-PEPS du MS A, qui comprend le protocole STORK. Ce dernier va alors authentifier l'utilisateur.

- Le deuxième modèle est quant à lui, basé sur un middleware spécifique au SP proposant le service. Le middleware utilisé par le SP permet l'intégration des tokens eID étrangers. Ce modèle est appelé 'Middleware model' ou encore 'Decentralized Deployment Model.

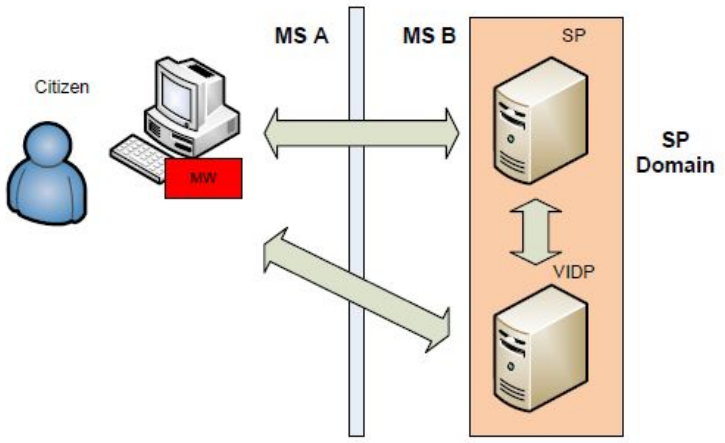

FIGURE 3.9 – Le modèle Middleware

Dans la figure présentée ci-dessus, nous distinguons un module V-IDP qui est un composant permettant l'interopérabilité entre les différents MS. Il est utilisé par les SPs et il intègre les différents types de tokens eID des états impliqués dans le projet STORK. Lors de l'accès au SP du MS B, la requête d'authentification sera gérée par le V-IDP, qui autorisera l'accès ou non à la ressource.

Il est intéressant de noter que ces deux modèles peuvent interagir ensemble sans aucun problème. Ceci nous donne alors quatre types d'interaction possibles entre deux MS. Le framework utilise les technologies SAML et les certificats.
SAML est utilisé dans son mode Web Browser SSO, permettant l'échange d'attribut via des assertions. Les certificats sont utilisés pour l'authentification via la carte d'identité. Ils impliquent la mise en place d'une PKI.
STORK permet de définir quatre processus métier de haut niveau. Un cas d'utilisation est disponible en annexe STORK 2.0.

- Le process 'Authentication on behalf of' permet à l'utilisateur d'accéder aux données de la personne représentée. Elles sont transmises au SP qui va les associer à la représentation qu'il en a. Cela veut dire qu'il peut voir la personne comme par exemple un étudiant, une médecin ou encore un partenaire. Il va donc identifier la relation entre l'utilisateur et lui-même. Ce process utilise la notion de mandat. Grâce à ce mandat, une personne peut représenter une autre personne ou une entité légale.
- Le processus 'Powers' intervient lors de signature digitale. Il permet au SP de vérifier que l'utilisateur possède assez de droits pour représenter la personne qu'il désire représenter. Ce processus se fait via la vérification de la signature de l'utilisateur par le SP.
- Le processus 'Business attributes' permet la transmission d'attributs métiers via une autorité centralisée.
- Le processus 'Powers Validation' permet le stockage des droits des utilisateurs auprès des SPs. Afin de les stocker, ils doivent vérifier au préalable qu'ils sont toujours valides.

Dans le cadre de ces flows et via SAML, le framework peut récupérer les attributs personnels de l'utilisateur mais aussi ceux de l'entité qu'il représente. Par exemple, un utilisateur belge qui travaille pour une société française pourrait se connecter avec sa carte d'identité en Espagne et représenter une personne morale, qui est sa société.

D'un point de vue sécurité, ce système possède plusieurs avantages. Sachant qu'il est basé sur SAML et les certificats, il possèdera les avantages de sécurité de ces deux éléments. L'aspect protection des données utilisateurs n'est pas mis de côté. Le framework est orienté vers une approche 'User-Centric', permettant la bonne gestion de leur données personnelles. Chaque échange de données avec un SP impliquera un consentement de l'utilisateur.
Un dernier attribut de sécurité est que le framework empêche de faire une corrélation entre l'ID et l'utilisateur final via une couche d'anonymisation.

3.2.5 CEF eID

Le projet STORK 2.0 s'est terminé fin de l'année 2015 et le nouveau projet CEF eID (aussi appelé eID Building Blocks) est son successeur[Hoffmann, 2015]. L'idée derrière ceci est exactement la même que celle du projet STORK, c'est à dire, promovoir l'interopérabilité entre services de différents pays de l'UE. Par rapport au projet STORK 2.0, CEF eID apporte en plus, un support, des formations , de la documentation ainsi que des modules Open Source développés. Le projet est lié à la régulation quant aux transactions et aux signatures électroniques , eIDAS. Cette régulation a été mise en place pour permettre les échanges électroniques entre différents stakeholders tels que les citoyens, les administrations publiques ou business privés. Cela a pu permettre de mettre en place l'authentification eID entre services de différents pays ainsi qu'une armonisation du statut légal des signatures électroniques. Le schéma en annexe CEF-eID montre l'évolution du projet.

Les Buildings Blocks font partie d'un des progammes de la Commission Européenne permettant la mise en place services interopérables à travers les frontières. On appelle ces services, des DSI's.
Nous distinguons deux types de DSIs :

- Les 'Building block DSIs' qui sont des services (re)utilisables par plusieurs secteurs. Nous pourrions par exemple avoir des mêmes DSIs pour l'eau, le gaz et l'electricité. Ils peuvent aussi être ainsi intégrés des DSIs de plus grande envergure.
- Les 'Sector-specific DSIs' qui sont seulement limités à un seul et unique secteur.

Chaque DSI, qu'elle soit du premier ou du second type est composée de deux couches :

- La première couche, nommée 'Core Service Platforms' est implémentée et gérée par la commission. Elle permet la connectivité entre les différents MS.
- Le seconde couche , nommée 'Generic Services' permet d'établir le lien entre l'infrastructure nationale des MS et la première couche. Cette couche est entièrement implémentée et gérée par le MS.

Actuellement la commission met à disposition cinq types de Buildings Blocks DSI :

- Le block 'eID' permet aux administrations publiques et privées d'étendre l'utilisation de leurs services online aux citoyens de l'UE.
- Le block 'eSignature' permet aux administration et business d'accélérer la création et la vérification de signatures électroniques.
- Le block 'eDelivery' permet d'échanger des documents et des données entre administrations et business, de manière sécurisée.
- Le block 'eInvoicing' permet aux administrations de réaliser plus facilement de l'e-facturation en suivant les directives légales européennes.

- Le block 'Automated Translation' permet aux administrations de s'abtraire de la barrière linguistique lors d'échanges d'informations.

Chaque MS qui utilise CEF eID possède un noeud eIDAS. Un eIDAS-node peut avoir deux rôles différents. Le type de rôle dépend de l'origine de la requête reçue . Le premier est l'eIDAS-Node Connector. Un noeud est défini comme tel, lorsqu'il se situe dans le SP du MS. Il aura pour but de recevoir la requête d'authentification et la forwarder au noeud eIDAS du pays auquel le citoyen appartient. Le second est l'eIDAS-Node Proxy-Service. Ce rôle est assumé quand le noeud se trouve dans le pays du citoyen. Son but est de recevoir les requêtes d'authentification d'un autre noeud eIDAS, comme par exemple d'un eIDAS-Node Connector. L'eIDAS-Node Proxy-Service possède une interface avec l'infrastructure eID nationale, qui lui permet de démarrer une phase d'authentification et/ou d'identification pour un citoyen, au niveau de l'IdP ou de l'AP.

Les normes de sécurité sont les mêmes qu'évoquées pour STORK 2.0. Même si la documentation parle très peu du SSO pour le framework, sachant qu'il est basé sur SAML et son profil Web Browser SSO, il va donc de soi qu'il peut être utilisé dans ce mode de fonctionnement. Il permet d'avoir un MDSSO sécurisé.

3.3 L'infrastructure

Lors de la description des SSO, nous avons vu qu'ils pouvaient fonctionner différemment. Certains gèrent les identités tandis que d'autres non. Il en va de même pour l'authentification, certains ont un seul IdP possible par rapport aux autres qui peuvent en posséder plusieurs. Nous allons discuter de ces points dans cette section.

L'authentification SSO est divisée en deux groupes. D'un côté, nous avons l'authentification de type centralisée et de l'autre, celle qui est décentralisée.
La première permet d'avoir un point d'entrée central par lequel toutes les requêtes d'authentification vont passer. Les infrastructures basées sur CAS voient leur architecture orientée dans ce sens. Ils ne possèdent qu'un seul IdP que les SPs clients interrogent afin de connaitre le statut de l'authentification pour un utilisateur.
A contriaro, les systèmes basés sur l'authentification décentralisée permettent d'avoir du SSO via plusieurs IdPs. C'est le cas d'OpenID et OAuth. Les SPs qui utilisent ces protocoles doivent posséder une outil de découverte d'IdP, leur permettant de savoir quel IdP a réalisé l'authentification de l'utilisateur.
SAML lui peut fonctionner dans les deux modes. Soit n'avoir qu'un seul IdP, soit en avoir plusieurs comme c'est le cas de CEF eID, où les SPs peuvent recevoir des assertions des différents IdPs impliqués dans la fédération SAML.
Lorsque nous parlons d'authentification centralisée, il est adéquat de distinguer comment le flux démarre. Si le flux démarre directement de l'IdP, alors nous pouvons dire que l'authentification est centralisée en

un seul point. Une fois la phase réalisée, l'utilisateur pourra cliquer sur un lien pour accéder à une ou plusieurs application(s) impliquée(s) dans le SSO. Par contre, si nous démarrons le flux au niveau de l'application, cette dernière va alors rediriger l'utilisateur vers l'IdP. Dans ce cas ci, le point d'entrée est l'application elle-même. Dès lors, sachant qu'elle n'effectue pas l'authentification, nous pourrions que nous sommes en présence d'un système décentralisé.
De la même façon pour un système décentralisé, un utilisateur n'accédant aux applications que via un seul et unique IdP pourrait se dire qu'il se trouve en présence d'un système centralisé. Or ce n'est pas le cas. Nous pouvons donc émettre les deux hypothèses suivantes :

- Si nous nous trouvons dans une situation où le système est dit centralisé, nous démarrons la phase d'authentification au niveau de l'IdP.
- Si nous nous trouvons dans une situation où le système est dit décentralisé, nous démarrons la phase au niveau de l'application et l'authentification pourrait se faire via plusieurs IdPs différents.

Nous remarquons quand même que les implémentations de ces protocoles ne suivent pas toujours cette règle. Sachant qu'elles rajoutent des fonctionnalités, l'architecture du SSO peut donc évoluer. C'est le cas de Facebook Connect, avec sa couche d'authentification. Si à la base, OAuth est décentralisé, ce n'est pas le cas de Facebook Connect. L'IdP dans ce cas est unique et se situe au niveau du serveur d'authentification Facebook.

Il est évident que le choix d'une architecture dont l'authentification est centralisée ou décentralisée, n'est pas toujours simple. Selon nous, il est intéressant d'avoir une base d'analyse via un tableau comparatif reprenant les avantages et les inconvénients de chacune.

TABLE 3.1 – Authentification centralisée versus décentralisée

	Avantages	**Inconvénients**
Centralisée	Politique de sécurité unique Relation de confiance unique avec les SPs Gestion des identités simplifiée Coûts réduits	L'IdP est un SPOF Redondance souvent nécessaire
Décentralisée	Plusieurs IdPs Implémentation IdP possible pour tout le monde	IdPs parfois mal sécurisés ou malicieux Coûts augmentés

Nous allons maintenant détailler ce tableau :

- **AUTHENTIFICATION CENTRALISÉE**
  1. **AVANTAGES :**
     (a) La mise en place d'une politique de sécurité globale permet d'avoir le même niveau de sécurité au niveau de l'authentification.
     (b) Les applications impliquées dans le SSO ne devront mettre en place qu'une seule relation

de confiance avec un seul IdP.

(c) La gestion des identités des utilisateurs est simplifiée par cette centralisation.

(d) Les trois éléments cités juste avant ont un impact sur les coûts. Le fait de regrouper l'ensemble de ces activités en un seul point diminue les coûts de maintenance et de déploiement de l'infrastructure.

2. **Inconvénients :**

(a) Un seul IdP est un SPOF. S'il tombe ou est compromis, alors, plus personne ne pourra bénécifier du SSO.

(b) Le premier inconvénient entraine généralement une redondance de l'infrastructure en cas de panne ou d'attaque. La mise en place d'outils comme des loadbalancers est à envisager dans le cadre d'un centralisation de l'authentification. Cela entraine certains coûts supplémentaires si cette option est prise.

- **Authentification décentralisée**

1. **Avantages :**

(a) L'avantage des systèmes décentralisés est qu'ils peuvent comporter plusieurs IdPs. Ceci est un avantage non négligable en cas de panne.

(b) Dans le cadre de OAuth 2.0 et OIDC, tout le monde peut avoir son site agissant comme IdP. Ceci a l'avantage de ne pas lier le business à de grands acteurs comme Google, Facebook ou Twitter.

2. **Inconvénients :**

(a) Sachant que chacun peut avoir son IdP, il peut être mal sécurisé ou malicieux. Aucune politique de sécurité n'agit globalement sur eux.

(b) Le premier argument fait en sorte que des configurations, serveurs, applications sont généralement dupliquées pour chaque IdP. Cela entraine des coûts supplémentaires.

Les SSO et la gestion des identités sont souvent liés. Le principe de la gestion des identités est de reconnaitre l'utilisateur afin de lui attribuer certains droits spécifiques. Celle-ci est importante lorsqu'elle est couplée à la phase d'authentification. Par mesure de précision, nous voulons dire que l'utilisateur s'authentifie en premier puis se voit attribuer des droits. Ce sont donc deux phases différentes mais complémentaires.

Nous distinguons plusieurs moyens de le faire. Le premier moyen est la gestion des identités de façon locale, au sein même de l'application. Si ce moyen est généralement choisi d'un cadre assez simpliste, où le déploiement doit être rapide et efficace, nous nous voyons très vite confronter à des problèmes en terme de centralisation des profils et en terme d'intégration avec d'autres systèmes. Ceci laisse peu de place à l'évolution de l'application.
Dès lors, le deuxième moyen est un système basé sur des annuaires comme le LDAP ou encore l'AD. Les

applications sont configurées de telle sorte à interroger ces annuaires lors de la phase d'authentification. Ils peuvent ajouter une certaine granularité en faisant une ségrégation par groupe , role , profil ou encore attribut. Les applications se baseront alors sur ceux-ci afin de définir des droits. Malgré cela, la gestion des groupes doit quand même être définie dans l'application. Les groupes récupérés de l'annuaire n'ont aucune valeur si l'application n'a pas défini de politique interne.
Le troisième moyen est un software ou un framework qui permet la gestion pointue des identités, appelé IAM. La différence par rapport à la méthode précédente est que dans ce cas-ci, l'application ne définit aucun droit mais les reçoit du composant IAM. Cela permet de découpler totalement la gestion des droits de l'application, et de la déléguer.
Ce type d'architecture est généralement plus complexe et implique des standards qui soient respectés. Elle est souvent mise en place par des entreprises possèdant des standards rigoureux.

Les SSO qui ont la possibilité de faire de la gestion d'identité font partie de la famille des True-SSO [11]. Sachant que les Pseudo-SSO rejouent l'authentification, les identitées des utilisateurs et donc leurs droits sont gérés soit par les applications elles-mêmes, soit par un autre module externe.
Par contre, nous remarquons que SAML est conçu à cet effet. C'est donc aussi le cas de ses implémentations vu qu'elles héritent des fonctionnalités de ceux-ci. OIDC et OAuth 2.0 possèdent plutôt une gestion de profils utilisateurs que de droits en soi. CAS, dans sa version de base, ne gère pas ceci. Cependant, certaines de ses implémentations le peuvent, c'est le cas du framework ECAS.

3.4 Comparaison des SSO

Dans cette section, nous allons différencier les SSO de par leurs avantages et leurs inconvenients. Nous commencerons pour comparer les True et Pseudo SSO ainsi que les Local et Proxy-based SSO. Après cela, dans le cadre des True-SSO, nous ferons une comparaison des quatre protocoles que nous avons expliqués ci-dessus.

3.4.1 Les quatres types de SSO

Les True et les Pseudo SSO possèdent chacun des avantages. Le choix d'implémenter l'une ou l'autre sorte dépend de différents critères. Le coût, le déploiement , la maintenance, l'infrastructure, les contraintes légales [12] sont des exemples d'arguments qui peuvent faire pencher la balance.
Nous pourrions créer un tableau nous-même, cependant nous n'allons pas réinventer la roue alors qu'il en existe déjà un dans la littérature :

11. Une exception est possible mais nous n'en avons pas vue lors de notre recherche.
12. Pour donner une illustration, dans le cadre de la sécurité sociale, les guichets d'entreprise ont l'obligation de mettre en place une authentification par carte d'identité.

	Local pseudo-SSO	*Proxy-based pseudo-SSO*	*Local true-SSO*	*Proxy-based true SSO*
Pseudonymity and Unlinkability	cannot be guaranteed	cannot be guaranteed	can be guaranteed	can be guaranteed
Anonymous Network Access	needs additional services	can be integrated	needs additional services	can be integrated
Support for User Mobility	needs additional services	under suitable authentication method	needs additional services	under suitable authentication method
Use in Untrusted Environment	not supported	under suitable authentication method	not supported	under suitable authentication method
Deployment Costs	low	low	high	high
Maintenance Costs	potentially high	potentially high	low	low
Running Costs	low	high	low	high
Trust Relationships	dynamically changing	dynamically changing	concrete and consistent	concrete and consistent

FIGURE 3.10 – Tableau comparatif des SSO[Pashalidis and Mitchell, 2003]

L'explication de ce tableau est la suivante :

- Dans le but de respecter l'aspect privé des données des utilisateurs, la pseudonimité couplée avec la non-liaison d'une identité peuvent être utilisées. L'idée derrière ces termes est d'utiliser un mécanisme de pseudonyme (John Doe pourrait être représenté par le pseudonyme 'user21') à travers le SSO afin de ne pas lier un identifiant à un utilisateur concret. Nous voyons ce type d'architecture si des données personnelles sont impliquées dans le SSO, comme des adresses email pouvant être utilisées comme identifiant principal.

  De par leur fonctionnement, les Pseudo-SSO ne peuvent pas disposer de ces qualités par le fait que les identités sont gérées au niveau des SPs. Par contre, sachant que les identités peuvent être gérées avec les True-SSO, ces qualités sont généralement adoptées dans ce cadre d'utilisation.
- L'anonymité de l'accès réseau est une qualité complémentaire aux deux premières que nous venons de présenter. Si certaines couches réseaux sont mal sécurisées, un attaquant aura la possibilité de faire une corrélation entre un utilisateur connecté et une identité. Ceci peut être réalisé par la localisation géographique de l'utilisateur. Les trames réseaux peuvent donner beaucoup d'informations à ce propos.

  Cette qualité est possible via une couche supplémentaire agissant en tant que couche d'anonymisation, comme le font certains proxies. Que l'on soit dans un Pseudo ou un True-SSO, les sous-familles Proxy-based permettent de mettre en oeuvre cette couche. Un attaquant se plaçant au niveau du SP ne verra que l'IP du proxy.

Les SSO locaux ne possèdent pas cette qualité implicitement. L'ajout de composants externes est nécessaire afin de pouvoir en disposer.

- Le support de la mobilité de l'utilisateur correspond au fait que le système peut être déplacé facilement. Si pour les Proxy-based SSO, ceci est assez simple, grâce à un changement de configuration, cela est plus dur pour les SSO locaux. Il faut généralement des composants ou des architectures particulières. Par exemple, une base de données contenant les identifiants qui serait externalisée fonctionnerait. Les identifiants seraient téléchargés par le composant SSO lors de l'authentification.
- L'utilisation du composant SSO dans un environnement de confiance est à prendre en compte. Certains environnements comme les WIFI ouverts, les cybercafés restent peu sécurisés et sujets à plusieurs types d'attaques afin de voler les identifiants ou les données personnelles. Les SSO de type Proxy-based peuvent pallier à ce manque de sécurité dans ces environnements. Partons quand même du principe que l'interaction entre l'utilisateur et le proxy est le point faible de la chaine. Dès lors, une authentification plus forte comme un code OTP ou un certificat pourrait remédier à cela.

  Le SSO local doit se trouver dans un environnement sécurité, sinon il est inutile de le mettre en place dans de telles conditions.
- Les coûts de déploiement sont plus applicables aux SSO de type True-SSO, qu'ils soient en mode Local ou Proxy-based. Nous avons déjà évoqué les raisons dans l'état de l'art, nous ne reviendrons pas dessus.
- Les coûts de maintenance, à contrario s'appliquent aux SSO de type Pseudo-SSO pour des raisons déjà citées.
- Les coûts de fonctionnement représentent les coûts du SSO pendant qu'il tourne. Ceux des SSO locaux sont faibles puisque une fois la machine éteinte après utilisation, il n'y a plus rien qui tourne. Ceux des Proxy-based SSO sont plus élevés par le fait que la machine fonctionne en permanence et est en interaction avec les SPs. Cependant, de notre avis, avec les infrastructures virtualisées qui sont omniprésentes de nos jours, ce type de coût reste quand même très faible.
- Les relations de confiance entre SPs et IdPs sont plus simples dans un environnement où le SSO est local, vu que tout l'environnement doit être de confiance.

  Il faut quand même distinguer la relation de confiance dans le cadre des True et Pseudo-SSO.

  Dans un True-SSO, la relation est bien définie. Le SP et l'IdP se connaissent entre eux et effectuent leurs échanges de manière sécurisée. Ceci est réalisé via une infrastructure bien définie.

  Pour les Pseudo-SSO, c'est quand même différent. Le SP ne connait pas l'IdP, mais le contraire bien. L'utilisateur qui se connecte devra alors faire pleinement confiance au composant SSO. Le composant connait le SP et sa méthode d'authentification. Si elle venait à évoluer, il devrait en être de même dans sa configuration.

Il est évident que d'autres critères peuvent être ajoutés à ce tableau. Nous en retenons certains :

Le retour sur investissement :
Nous pourrions ajouter que dans le cadre d'une infrastructure utilisant un composant SSO, le retour sur investissement peut être plus rapide. En effet, si la société suit de bons standards de programmation au niveau de l'implementation de ses méthodes d'authentification, ces dernières ne devraient pas beaucoup différer. Nous aurions au plus entre cinq ou dix méthodes différentes à enregistrer au niveau du composant SSO. Cela permettrait de gagner un temps considérable et donc un gain d'argent. Un projet d'implémentation SSO qui devrait se dérouler sur quelques mois pourrait passer à quelques semaines [13].

L'interopabilité :
Ce critère est important dans la mesure où le système doit s'intégrer avec l'infrastructure actuelle. Pour les Pseudo-SSO, ceci est assez simple vu les SPs ne les connaissent pas. Cependant, pour les True-SSO, ceci peut être plus difficile. L'étude de l'intégration du système et des changements au niveau des SPs doit être établie. Certains True-SSO sont plus facilement intégrables que d'autres. Cela a bien sur un impact sur les coûts.
Nous verrons cela lors de la comparaison entre les protocoles des True-SSO.

Le type d'environnement :
En fonction du type d'environnement dans lequel nous nous trouvons, l'impact sur la sécurité est différent. Si nous nous trouvons dans un environnement fermé, certains attributs de sécurité sont moins importants. Typiquement les attributs de pseudonimité et de non-liaison sont souvent ignorés. Les seuls utilisateurs ayant accès au réseau sont les utilisateurs internes. Cela minimise les attaques qu'il pourrait y avoir et le vol de données privées. Nous nous retrouvons plutôt alors dans des coûts classiques de déploiement, de maintenance et fonctionnels[Pashalidis and Mitchell, 2003].
Dans un environnement ouvert, la sécurité est importante. Il faut veiller à beaucoup plus de critères, voire même des critères légaux. L'aspect privacy décrit ci-dessus doit être pris en compte mais aussi celui de la redondance. Un système ouvert est souvent accédé par un nombre d'utilisateurs plus important qu'un système fermé. Cela entraine la mise en place d'une infrastructure différente. De ce fait, nous voyons souvent des implémentations de type True-SSO pour ces environnements ouverts.
A titre d'exemple, dans notre expérience professionelle, nous avons déjà rencontré ce type de cas :

- En interne, le SSO était présenté avec un serveur SSO, un portail, un noeud par application.
- En externe, nous avions deux reverse proxies, un loadbalancer, deux serveurs SSO, le portail externe répliqué et clusterisé et plusieurs noeuds par application.

Il est évident que cette redondance fait aussi partie de la sécurité.
Au final, nous nous retrouvons avec les mêmes coûts classiques qu'un environnement fermé mais avec

13. Temps à adapter en fonction de l'envergure du projet. Ceci est approximatif.

des coûts additionnels pour les différents critères que nous venons de citer. Si nous devons prendre une échelle, le coût total d'un environnement ouvert pourrait être deux à trois fois plus élevé que celui d'un environnement fermé.

3.4.2 Les quatres protocoles True-SSO

Une dernière comparaison a été réalisée. Celle-ci concerne les quatre protocoles liés à un True-SSO. Nous avons créé un tableau comparatif entre ceux-ci sur base de différents critères :

TABLE 3.2 – Comparaison des protocoles SSO

	CAS	**OpenID Connect**	**OAuth 2.0**	**SAML 2.0**
Format des messages	Ticket opaque	JSON	JSON	XML
Contrôle de l'IdP	oui	non	non	oui
Sécurité des échanges	HTTPS	HTTPS JWE JWS	HTTPS*	HTTPS Signature XML Encryption XML
Contexte	Portail Webmail	Partenariat Echanges sécurisés Fédération d'identités	Délégation de droits	Partenariat Echanges sécurisés Fédération d'identités
Complexité	Faible	Faible	Faible	Elevée
Coût	Faible	Faible	Faible	Elevé
Interopérabilité	Difficile	Forte	Difficile	Forte

Selon nous, ces critères sélectionnés sont les plus adéquats pour comparer les quatre protocoles. L'étoile '*' indique que le paramètre est soumis à une exception.
Les voici :

- Format message : ce critère indique le format des messages qui est utilisé lors d'un échange SSO entre SP et IdP.
- Contrôle IdP : ce critère indique si les échanges entre SP et IdP sont contrôlés.
- Sécurité échanges : ce critère représente le niveau de sécurité mis en place lors des échanges des messages entre SP et IdP.
- Contexte : ce critère représente le contexte d'utilisation dans lequel le protocole est utilisé. Il est évident que ce soit des contextes généraux et qu'il peut être appliqué à un contexte similaire.
- Complexité : ce critère représente la complexité de la mise en place du protocole.
- Coût : ce critère représente le coût total de la solution. Il reprend les coûts fonctionnels, de déploiement, de maintenance ainsi que les éventuels coûts liés à l'environnement dans lequel le protocole est mis en place.
- Interopérabilité : ce critère représente la façon dans le SSO peut être intégré avec l'infrastructure actuelle.

- Environnement : ce critère représente si l'environnement est ouvert ou fermé.

Les différents critères utilisés dans le tableau vont être expliqués ci-dessous :

— **Format message**

CAS utilise des tickets opaques. Ces derniers sont stockés dans le navigateur lors d'une session SSO et permettent de garder la session active. Les SPs vérifient leur présence.

OIDC et OAuth 2.0 utilisent le format JSON lors des échanges entre IdPs et SPs. Ces messages sont interceptés par les SPs et parsés, afin de récupérer les informations nécessaires. Ceci inclut, les autorisations ainsi que les informations potentielles concernant l'utilisateur dans le cas d'OIDC.

SAML utilise le format XML lors du transfert d'informations des IdPs vers les SPs.

— **Contrôle IdP**

Lors d'un échange SSO CAS, il y aura une vérification du serveur duquel vient le ticket. En cas d'ursurpation, sachant que le SP connait le serveur CAS, s'il ne retrouve pas ce dernier, l'accès sera rejeté.

OIDC et OAuth 2.0 n'utilisent pas de ce mécanisme de vérification de la provenance du message. Un IdP pourrait envoyer des messages malicieux aux différents SPs.

SAML permet aux SPs de vérifier que le message provient bien du bon IdP.

— **Sécurité échanges**

La sécurité des échanges entre SPs et IdPs utilisent souvent des mécanismes cryptographiques ou de signature de messages.

Même si CAS ne possède pas de mécanisme cryptographique de base , ses échanges doivent se faire en HTTPS. Son traffic est donc chiffré.

OIDC et SAML possèdent aussi des mécanismes cryptographiques. Ils permettent le chiffrement des messages (JWE et XML encryption) et celui des échanges via HTTPS. Afin d'assurer la sécurité de l'intégrité des messages, la signature des messages peut aussi se faire. De plus, sachant qu'ils envoient tous deux des informations concernant l'utilisateur, le consentement peut être demandé. OIDC possède cet attribut dans son package de base. SAML nécessite une configuration spécifique afin de pouvoir le faire.

OAuth ne possède aucun mécanisme cryptographique. De plus l'utilisation du protocole HTTPS n'est pas obligatoire lors des échanges. Cependant il est quand même vivement conseillé de le mettre en place.

— **Contexte**

Le contexte d'utilisation dans lequel CAS est utilisé est souvent des portails de liens ou Webmail. Ce système est très simple à mettre en place au niveau applicatif. De par son mode proxy, le portail serait l'application proxy qui distribuerait les tickets pour les applications présentes derrière celui-ci.

OIDC est actuellement plutôt utilisé par ses utilisateurs dans un mode de facilitation de surf sur Internet, comme par exemple du SSO entre une application tierce et un réseau social. Beaucoup

d'utilisateurs sont friands de ce SSO, leur permettant de lier leur compte Microsoft, Gmail, Dropbox, Facebook etc... à leur application. De plus dans une optique de développement, il permet aux développeurs de ne plus s'occuper de la phase d'authentification et de déléguer celle-ci à un provider OpenID de confiance.

Même si OIDC est souvent utilisé avec des réseaux sociaux, rien n'empêche de l'utiliser dans le cadre d'une entreprise qui voudrait faire de la fédération d'identité. En effet, sachant qu'il possède les mêmes qualités de sécurité que SAML, il peut aussi être utilisé dans les mêmes cas d'utilisation.

OAuth 2.0, quant à lui, est plutôt dans une optique de vérificateur d'identité chez les IdPs. Il est utilisé dans la majeure partie des cas avec des réseaux sociaux. Cependant, rien n'empêche de l'utiliser autrement, mais si selon, cela reste assez limité par le fait qu'il ne gère que les autorisations.

SAML est un SSO permettant le SSO entre domaines. Il est très utilisé dans un cadre de partenariat entre entités. Pour prendre un exemple concret, le but de CEF eID est de faire un partenariat entre tous les pays de l'UE. Il est très utilisé lors d'échanges d'informations de façon sécurisée. De plus, sachant que SAML permet la fédération d'identités, certains sociétés mixtent Web Browser SSO avec celle-ci, permettant ainsi une centralisation des identités à travers tous les SPs.

— **Complexité**

La complexité de CAS, OIDC et OAuth est assez faible. Les implémentations de celles-ci sont assez rapides et ne nécessitent pas un grand changement d'infrastructure.

Ce n'est pas le cas de SAML. Ce dernier est complexe et nécessite une bonne analyse avant sa mise en place. Ceci est explicable car un partenariat entre différentes sociétés est toujours assez complexes et est soumis à certaines contraintes.

— **Coût**

De manière générale, le coût fonctionnels des quatre protocoles est assez égal. Ils doivent tous posséder au minimum un IdP disponible continuellement. CAS est peut-être plus sensible à cela par rapport aux autres car il ne possède qu'un seul IdP.

Sachant que ce sont tous les quatre des True-SSO, les coûts de déploiement et de maintenance sont assez semblables. Cependant l'implémentation de SAML peut devenir coûteuse quand il est exposé dans un environnement ouvert. Les contraintes de sécurité et de redondance peuvent très vite faire augmenter le coût total du projet.

— **Interopérabilité**

CAS est difficilement interopérable. Lorsqu'il est mis en place dans une infrastructure, les SPs doivent être adaptés en fonction. C'est surtout le cas dans son mode proxy, où une application proxy joue le rôle de relai pour l'IdP.

OAuth est difficilement interopérable comme il est indiqué dans leur docmentation[D. Hardt, 2012]. Il est plutôt considéré comme un framework avec différents composants que nous pouvons ajouter. Il est donc difficile de l'intégrer dans un système existant.

OIDC et SAML sont quand à eux facilement interopérable dans une infrastructure existante. Ils peuvent s'y ajouter moyennant quelques configurations supplémentaires. OIDC est intéressant car

il est totalement compatible avec une infrastructure OAuth 2.0. Dans ce cas, son intégration peut se faire très rapidement.

— **Environnement**

CAS est un SSO qui est plutôt utilisé dans un environnement fermé de par son utilisation.

OIDC et OAuth sont quand à eux plutôt orientés environnement ouvert du fait qu'ils sont indépendant des IdPs et que leur utilisation est souvent axée vers les réseaux sociaux. Cependant rien n'empêche de les utiliser dans un environnement fermé[14].

SAML est quand à lui utilisable au bon vouloir, que ce soit dans un environnement fermé ou ouvert. Il est facilement adaptable à toute utilisation, complexe ou non.

A travers ces comparaisons, nous pouvons voir que deux protocoles se ressemblent fortement de par leur but, leurs fonctionnalités et leur niveau de sécurité. Ce sont OIDC et SAML. Ils utilisent des mécanismes de sécurité différents par la technologie mais similaires au niveau du résultat. De plus, dans le cadre du SSO, tous deux permettent une fédération d'identité entre domaines. Il existe quand même des différences qui sont selon nous mineures. Elles sont disponibles à travers le tableau comparatif en annexe SAML vs OpenID.

L'auteur qui a réalisé ce tableau[(Novay), 2012] effectue sa comparaison en les différenciant sur plusieurs points :

- Le format des messages est différent 'XML / SOAP' versus 'JSON / REST'. Cependant ceci n'a pas grande importance sur le fond. Seules les librairies utilisées pour utiliser ces technologies peuvent être différentes et donc plus lourdes. Néanmoins JSON semble être plus léger et plus simple à implémenter. De plus, par rapport à XML, JSON s'utilise facilement avec du Javascript.
- OIDC utilise un canal de communication en back-end, une fois le token d'accès délivré. SAML fait transister ses messages via le navigateur, aussi appelé front-end. Le navigateur est plus soumis à des attaques de phishing que le canal back-end.
- Il voit une différence entre OIDC qui est 'user-centric' tandis que SAML est 'organisation-centric'. Cependant il décrit aussi que SAML peut basculer dans un mode user-centric. Du point de l'utilisateur, l'approche user-centric est meilleure pour le contrôle de ses données.
- La sécurisation des messages (chiffrement et signature) n'est pas obligatoire pour ces deux protocoles. OIDC force les flux d'informations entre IdPs et SPs à utiliser un canal sécurisé comme TLS, tandis que SAML ne l'oblige pas.
- Les relations de confiance établies entre SPs et IdPs dans un flux SAML sont statiques. Chaque nouveau SP qui doit être ajouté impliquera la modification de la configuration de l'IdP. OIDC permet sa fonctionnalité de façon dynamique, en fournissant un token d'accès. Cet aspect est peut-être plus intéressant pour les réseaux sociaux qui voient de nouvelles applications développées se greffer à la fédération tous les jours. Il serait difficile de mettre ceci en place avec un enregistrement statique comme celui de SAML.

  Dans le cadre d'une entreprise hébergeant ses propres applications, ce paramètre est négligeable.

14. Surtout OpenID Connect.

Les deux protocoles proposent les mêmes fonctionnalités dans leur mode SSO. Globalement, les différences mineures entre eux ne sont pas énormes. Par contre, OIDC est intéressant par le fait qu'il puisse faire du SSO avec les applications mobiles natives, ce que SAML ne permet pas.

3.5 Réponse aux questions : Partie II

En ce début de chapitre, nous nous sommes interrogés sur la viabilité d'avoir un SSO global. Ceci dans le but d'éviter à l'utilisateur d'entrer son secret plusieurs fois, et donc renforcer la sécurité.
Nous remarquons très vite que ceci est peut-être possible en théorie [15] mais très peu en pratique. Les contraintes de ces outils d'authentification empêchent de le répandre complètement sur le Web. Que ce soit les Pseudo-SSO, impactés par tout changement de la méthode d'authentification de l'application ou encore les True-SSO qui lient l'application à leur méthode de fonctionnement, la diversité du Web fait que ceci est difficilement réalisable. Cependant, les recherches continuent et les protocoles s'améliorent. OIDC est en bonne voie. Il permet le SSO Web et non Web (application native sur mobile) et possèdent les mêmes qualités de sécurité que SAML, tout en étant moins complexe. Nous pouvons donc nous demander s'il ne va pas remplacer SAML un jour et si de nouveaux protocoles ne vont pas découler de ce dernier. Avoir un SSO sécurisé et facile à implémenter pourrait faire changer la vision de la communauté et donc voir les développements s'orienter vers ce type de solution.

15. Même si cela est utopiste.

CHAPITRE 4

Outil d'aide à la décision

Les chapitres précédents nous ont permis d'établir un listing des différentes méthodes [1] et protocoles d'authentification.
Sur base de ceux-ci, nous avons décidé de créer un outil d'aide à la décision. Ce dernier sera présenté sous la forme d'un arbre de décision[Caron, 2011]. Son but est de permettre à tout responsable informatique qui le lirait, de choisir une politique d'authentification adéquate pour son entreprise.
Afin de garder une certaine consistance sachant que nous avons différencié les méthodes des protocoles eux-mêmes, deux arbres distints ont été réalisés. Le premier permet de faire un choix parmis les méthodes, tandis que le second est orienté vers le choix d'un protocole.
Pour ce faire, nous avons dû choisir certains critères. Le domaine de l'authentification est vaste et il est évident qu'il existe une infinité de critères qui pourraient rentrer dans ces arbres.

4.1 Les arbres

Chacun des arbres suivants qui va être présenté ci-dessous utilise des formes dans sa construction. Pour une meilleure compréhension du lecteur, une légende a été créée :

- Les méthodes ou protocoles sont indiqués en vert dans un rectangle. Ce sont les feuilles de l'arbre et elles représentent la décision finale par rapport aux choix réalisés à travers l'arbre.
- Les questions sont indiquées en noir [2].
- Les choix sont indiqués sur les flèches en violet.
- Les flèches possèdent des attributs comme par exemple "sécurité++". Ceci indique que le choix effectué s'oriente dans ce cas-ci vers une décision finale ou la sécurité sera de mise.
- Certains attributs possèdent des spécificités "+", "-", "++", "–". Ceci indique que l'attribut est fortement(+), très fortement(++), faiblement(-) ou très faiblement(–) orienté dans ce sens. A titre d'exemple, utilisabilité++ veut dire que la solution finale sera très simple à utiliser pour les utilisateurs.

  Un "+" ou un "-" veut dire que l'attribut possède le comportement mais pas totalement.

1. Et implémentations dans le cadre SSO. Nous garderons le terme méthode par simplification.
2. Exception pour le sommet de l'arbre des protocoles.

- Dans l'arbre des protocoles, il y a la présence de deux flèches en pointillé. Cela réprésente le fait que les choix peuvent être complémentaires. Nous pouvons donc avoir un protocole SSO dont l'authentification principale est forte.

Afin de pouvoir rendre ces arbres lisibles, nous avons du abrégér certains termes, principalement sur les flèches.

- Le terme "env" est l'abréviation de "environnement".
- Le terme "auth" est l'abréviation de "authentification".
- Le terme "interop" est l'abréviation de "interopérabilité".

4.1.1 L'arbre des méthodes

Pour respecter l'ordre dans lequel nous avons présenté l'authentification dans ce mémoire, nous présenterons d'abord l'arbre des méthodes :

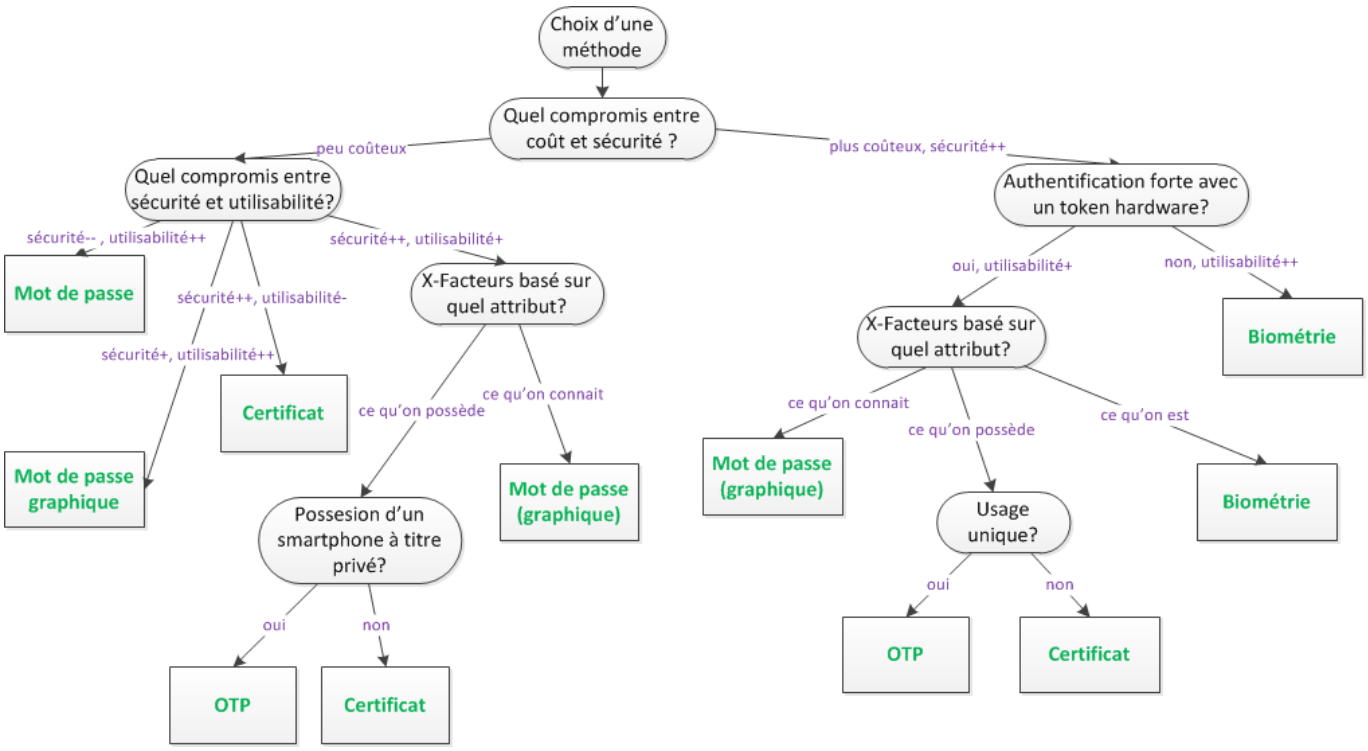

FIGURE 4.1 – Arbre de décision des méthodes

Cet arbre se base sur plusieurs critères que nous avons choisis. Si nous nous référons à la plupart des articles scientifiques qui parlent des méthodes d'authentification, trois critères principaux ressortent : le côut, la sécurité et l'utilisabilité.
Nous avons pensé à d'autres critères mais afin de pouvoir garder une certaine cohérence dans l'arbre, nous avons dû en éliminer.
La liste ci-dessous présente l'ensemble des critères et en couleur, ceux qui ont été retenus :

- **Le coût** est le nerf de la guerre. De notre opinion, c'est le critère le plus parlant pour un responsable informatique.
- **La sécurité** est aussi très importante. Tout le monde veut une sécurité maximale lors de la phase d'authentification. C'est encore plus le cas dans certains business comme le monde bancaire, militaire ou gouvernemental.
- **L'utilisabilité** peut faire qu'un système ne soit pas adopté par les utilisateurs finaux. Il doit donc être considéré au même titre que les deux critères précédents.
- **La possession d'un token hardware** fourni par l'entreprise doit être envisagée lors d'une authentification multi-facteurs.
- **La possession d'un smartphone privé** peut intervenir lors de la mise en place une authentification multi-facteurs.
- **L'authentification multi-facteurs basée sur ce que l'on est, sur ce que l'on possède ou sur ce que l'on connait** se réfèrent toutes trois à la définition de l'authentification forte.
- La facilité d'implémentation doit être évaluée. Il n'est pas toujours aisé pour les administrateurs systèmes d'implémenter une méthode ou une autre. Plusieurs d'entre elles pourraient être délaissées de par leur complexité de mise en place.
- Les contraintes légales sont parfois obligatoires. Dans le cadre de la Belgique, il existe des contraintes légales pour des entreprises, comme par exemple l'authentification par certificat (eID).
- La maintenance de la méthode peut parfois être lourde. Certaines méthodes comme les OTPs avec le renouvelement de la séquence ou les certificats avec leur PKI nécessitent une maintenance.
- La maturité de la méthode peut empêcher l'adoption de celle-ci. Nous voyons des méthodes délaissées par le fait qu'elles soient difficilement utilisables.

L'idée de choix derrière cet arbre est de voir quel compromis le responsable informatique est prêt à faire entre coût et la sécurité. L'utilisabilité sera évaluée après ces deux aspects.
Selon nous, le critère le plus parlant pour lui est le coût. Le choix de départ s'oriente dans ce sens et crée deux grands axes :

- Il peut s'orienter vers la branche de gauche, auquel cas, un compromis doit être fait entre sécurité et utilisabilité.

  Le premier choix s'oriente vers les mots de passe, dont la sécurité est faible mais l'utilisabilité très bonne.

  Le second choix est le mot de passe graphique. Cette méthode est un peu plus sécurisée que le mot de passe et possède une utilisabilité similaire à celle des mots de passe normaux.

  Le troisième choix est le certificat. Il offre une sécurité forte par rapport aux mots de passe (graphiques) mais son utilisabilité pour un utilisateur final est faible du fait qu'en cas d'erreur, il ne sait généralement pas quoi faire. Cependant il peut s'implémenter à moindre coût.

  Le dernier choix s'oriente vers une méthode de type multi-facteurs. Sa sécurité est élévée mais son utilisabilité est quand à elle, parfois contraignante. La sous-branche de gauche se base sur ce que l'utilisateur possède : un token ou un certicat. En partant de l'hypothèse que l'utilisateur

peut utiliser son smartphone privé et qu'il le possède en permanence sur lui, nous pouvons donc envisager un code temporaire OTP reçu sur son smartphone par SMS.

La sous-branche de droite se base sur ce que l'utilisateur connait, comme les mots de passe (graphiques). Pour garder une sécurité forte, il faut que le premier facteur soit différent des autres facteurs, car c'est souvent un mot de passe classique.

Nous pourrions ajouter une sous-branche supplémentaire basée sur ce que l'utilisateur est mais il nous semblait difficile d'avoir une méthode peu coûteuse offrant cette possibilité.

- Si nous revenons au premier niveau et que nous nous dirigeons vers la branche de droite, alors nous sommes en présence de systèmes dit plus coûteux mais disposant une sécurité accrue. C'est notamment le cas de la biométrie qui est directement à droite. Elle offre un niveau de sécurité élévé dû à la présence requise de l'individu ainsi qu'une utilisabilité très correcte par le fait que l'utilisateur n'a juste qu'à effectuer l'action biométrique demandée (empreinte digitale, vocale, scan de l'iris etc..). Cependant le coût de ce système est quand même élévé et reste limité à un certain type d'utilisateur et d'entreprise.

  La branche de gauche s'oriente vers le choix d'un système multi-facteurs avec un token hardware fourni par l'entreprise. Le coût global de ce système est élévé puisqu'un token est associé à une personne. Tout comme dans le cas du smartphone, ce système est parfois contraignant donc l'utilisabilité est diminuée (un seul +). Sachant que nous sommes en présence d'une authentification forte, nous avons trois choix. La sous-branche de gauche est basée sur ce que l'utilisateur connait : les mots de passe (graphiques). La sous-branche du milieu sur ce que l'utilisateur possède : OTP sur le device dans le cas d'un usage unique ou un certificat pouvant être réutilisé plusieurs fois. Et enfin la sous-branche de droite, représentée par la biométrie, qui est caractérisée par ce qu'il est.

4.1.2 L'arbre des protocoles

Le second arbre est celui des protocoles. Il est complémentaire à celui présenté ci-dessus, puisque les protocoles utilisent des méthodes d'authentification.

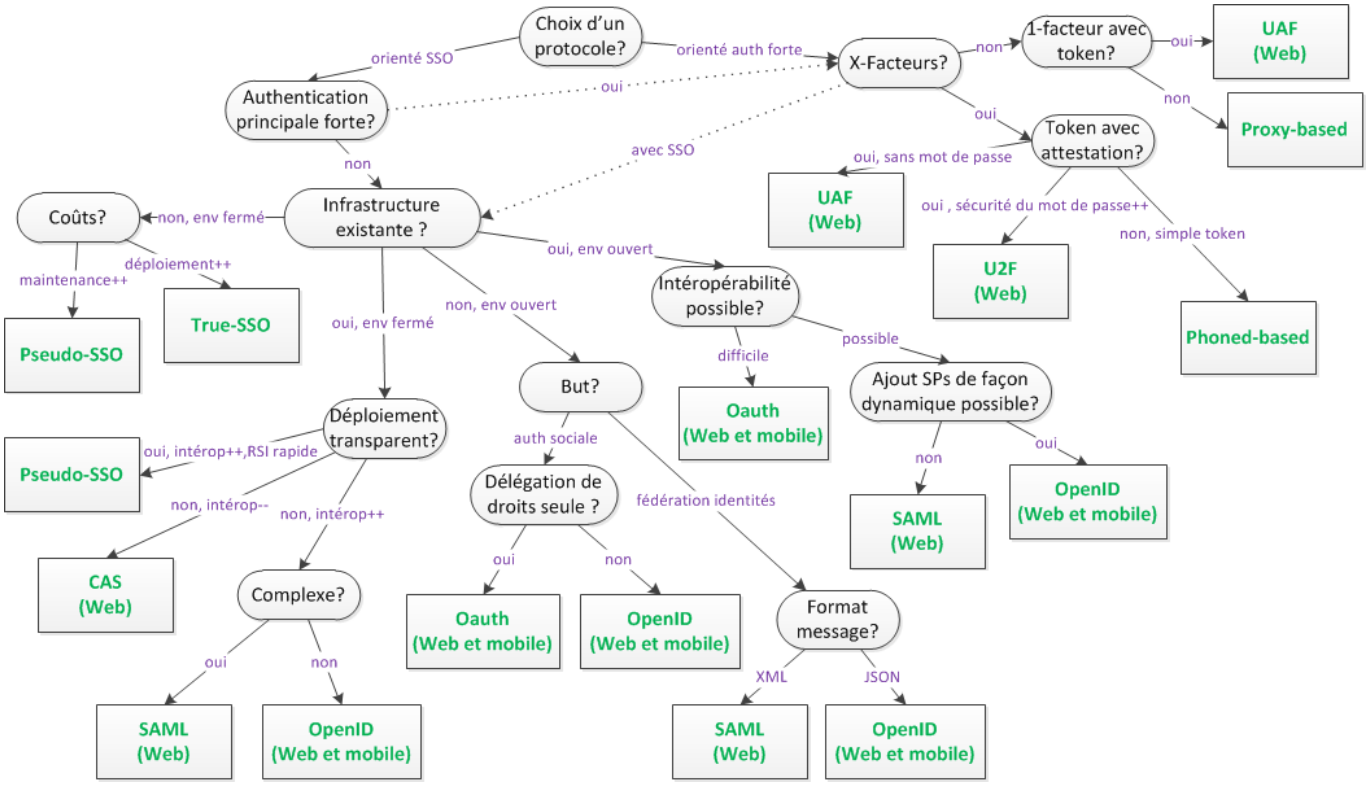

FIGURE 4.2 – Arbre de décision des protocoles

- **L'existance de l'infrastructure** est un point à prendre en compte afin de voir comment le protocole pour s'intégrer avec ou comment avoir un protocole adéquat lorsqu'elle n'existe pas.
- **L'interopérabilité** est lié au critère précédent. Certains protocoles ont plus de difficultés à être intégrés que d'autres dans une infrastructure existante.
- **L'environnement** peut être fermé ou ouvert. En fonction de ceci, certains aspects doivent être envisagés (loadbalacing, reserve proxy, firewall,...). Ceci entraine éventuellement des coûts supplémentaires.
- **Le coût** est aussi important. Dans le cadre du SSO, il y a des coûts fonctionnels, de déploiement ou de maintenance. Seuls ceux de maintenance et de déploiement sont pris en compte dans l'arbre.
- **Le RSI** [3] est un critère dont nous avions parlé lors des comparaisons entre SSO. Nous estimons que lors d'une implémentation, il ne doit pas être négligé.
- **La complexité** du système est un critère important. Certains protocoles sont plus complexes que d'autres lors de leur implémentation.
- **La transparence du déploiement** est à envisager. Il se peut qu'un responsable informatique ne veuille que très peu changer l'infrastruture existante, notamment au niveau des SPs.
- **Le but** du protocole permet d'avoir une vision plus claire lors d'une hésitation. Dans ce cadre, nous parlons d'authentification sociale ou de fédérations d'identités.
- **L'enregistrement des SPs** dans un cadre SSO est important. Il existe un SSO qui permet d'enregistrement dynamiquement les nouveaux SPs qui viennent s'ajouter.
- **Les facteurs d'authentification** comme un seul facteur ou le multi-facteurs sont à étudier

3. Retour sur investissement.

lors de l'implémentation d'un protocole. Lors d'un multi-facteur, il faut envisager l'utilisation de tokens. Certains protocoles renforcent l'authentification avec token par l'attestation de possession de celui-ci par son propriétaire. Un critère lié à l'authentification multi-facteurs est aussi de vouloir renforcer l'authentification principale par mot de passe via un second facteur.

- **L'orientation** Web, non Web et Mobile dépendent du type d'utilisateurs qui vont se connecter au système. Certains protocoles s'orientent uniquement vers le Web et/ou le mobile.
- Le nombre de failles lié au protocole est un critère qui pourrait éviter sa mise en place par le risque que cela engendrerait.
- Les contraintes légales sont tout comme pour les méthodes, un critère qui pourrait figer l'utilisation d'un protocole en fonction du business de l'entreprise. Ceci est plus le cas dans le cadre d'une méthode mais il ne faut pas l'exclure.
- L'indépendance a toute technologie est importante dans un environnement hétérogène. Certains protocoles sont liés à une technologie.
- La protection des données privées des utilisateurs est un atout lors du choix d'un protocole. De plus en plus de gens sont inquiets vis à vis de ceci et ne veulent pas voir leurs données répandues partout.

Tout comme l'arbre précédent, il existe deux grands axes dans cet arbre :

- Le responsable informatique peut s'orienter vers la branche de gauche où il devra choisir parmis les protocoles SSO. Cependant, il pourra envisager que l'authentification principale soit forte. C'est pour cela que le choix se divise en deux, permettant une complémentarité entre SSO et authentification forte.

  A partir de là, les sous-branches mettront en évidence la présence d'une infrastructure SSO déjà existante et dans ce cas, comment le protocole est interopérable avec celle-ci. Il faudra aussi analyser si le SSO sera destiné à un environnement ouvert ou fermé. Un SSO en environnement fermé est destiné aux employés internes de la société, tandis qu'un SSO en environnement ouvert est destiné aux clients et partenaires externes.

  Si l'environnement est fermé et qu'il n'existe pas d'infrastructure, alors nous avons le choix entre Pseudo-SSO et True-SSO, en fonction des coûts désirés.

  Si l'environnement est fermé et qu'il existe déjà une infrastructure, alors il faut envisager le type de déploiement voulu. Si nous voulons un déploiement transparent, alors il faudra s'orienter vers les Pseudo-SSO. Ils ont l'avantage de founir une interopabilité élevée ainsi qu'un retour sur un investissement très rapide. Si le déploiement n'est pas transparent et que l'interopabilité est faible, alors le choix est CAS. Sinon le dernier est un déploiement non transparent vu que les SPs doivent être modifiés mais cela s'oriente vers SAML ou OpenID, SAML étant plus complexe que son rival.

  Si l'environnement est ouvert et qu'il n'existe pas d'infrastructure, alors il faut envisager quel est le but final du SSO. Si nous voulons l'utiliser dans le cadre d'une authentification sociale, alors deux choix sont possibles. Le premier est OAuth, un framework qui n'est seulement basé que sur les autorisations. Le second, OpenID permet l'authentification et les autorisations. Si le but recherché est la fédération d'identités alors SAML et OpenID sont deux bons candidats. Leur différence se

fait au niveau du format des messages, XML et JSON. JSON reste quand même plus simple que XML et s'intégre très bien avec du Javascript.

Si l'environnement est ouvert et qu'il existe déjà une infrastructure, alors il faut voir comment il est possible d'intégrer le SSO. De ce point de vue, OAuth n'est pas un très bon candidat et son interopabilité est difficile. Par contre, SAML et OpenID le peuvent. Si le choix s'oriente dans ce sens, OpenID possède l'avantage de pouvoir ajouter dynamiquement les SPs, ce qui est très intéressant quand le nombre de SPs est conséquent.

- La branche de droite correspond aux protocoles utilisables dans le cadre d'une authentification forte.

  Si le choix est une authentification forte basée sur un seul facteur, alors il faut envisager la possession du token. UAF permet de s'authentifier localement sur ce token sans qu'il n'y ait un mot de passe à entrer du point de vue de l'expérience utilisateur. S'il n'y a pas de token, alors le protocole sera un proxy, agissant comme intermédiaire de sécurité entre le client et le SP.

  Par contre, si nous voulons un choix d'authentification forte avec multi-facteurs, il existe des protocoles proposant une authentification avec attestation de possession du token. C'est le cas de UAF et U2F. Le premier permet une authentification multi-facteurs sans avoir de mot de passe et le second permet le renforcement du mot de passe principal avec un second facteur. S'il n'y a pas d'attestation de possesion du token, alors le protocole basé sur les téléphones est le candidat voulu. Il permet de s'authentifier de manière sécurisée dans un environnement qui n'est pas de confiance.

4.2 Application de l'outil

4.2.1 Cas I : Unamur

Le premier cas est l'université Unamur. Le responsable IT était intéressé par les deux arbres.

1. Le cas d'utilisation présenté ici pour l'arbre des méthodes est l'application Webmail disponible pour les étudiants et les membres du personnel. Elle permet de consulter ses mails depuis un navigateur Web et est disponible en interne, comme en externe.

   Il a évoqué trois besoins :

   (a) L'authentification ne doit en aucun cas être un frein à l'utilisation de l'application.

   (b) La méthode utilisée doit être peu coûteuse.

   (c) La méthode d'authentification peut varier en fonction de l'adresse IP de l'utilisateur. Ceci est demandé dans le but de contrer les campagnes de phishing déjà rencontrées dans le passé. Si l'IP ne provient pas de Belgique, une authentification plus forte serait demandée.

   En parcourant l'arbre, il a bien rencontré le même choix que l'implémentation actuelle, c'est à dire le mot de passe. Cette méthode est peu coûteuse et utilisable par tous. Ce choix correspond aux deux premiers besoins.

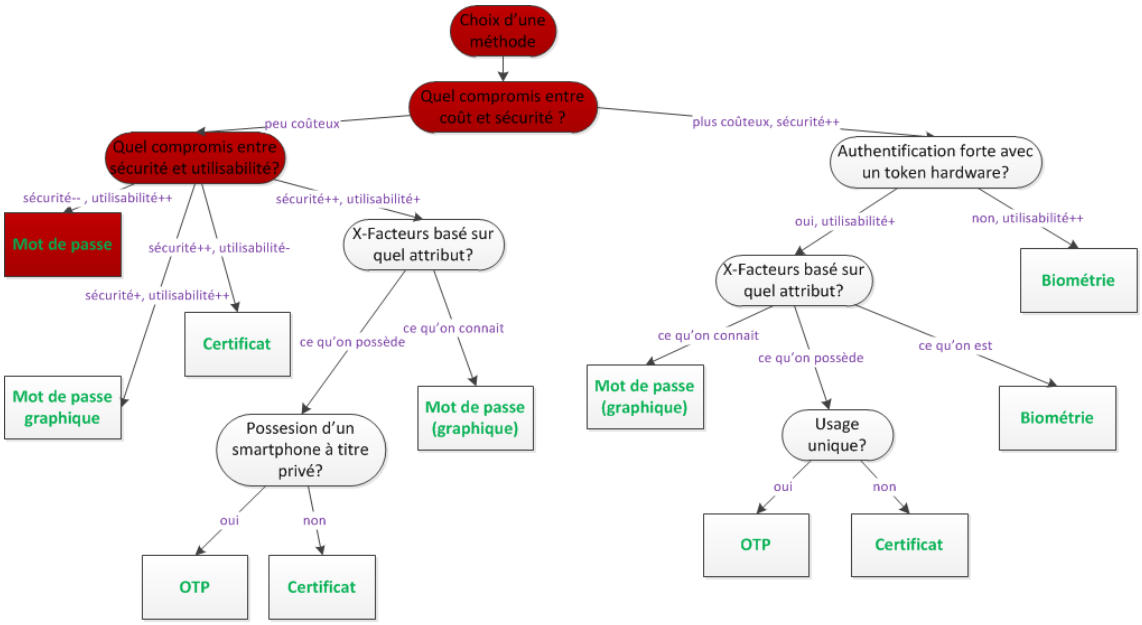

FIGURE 4.3 – Choix d'une méthode en fonction des deux premiers critères pour le cas UNamur

Le troisième besoin est actuellement à l'étude. L'Unamur a l'idée d'utiliser le canal SMS lors d'un second facteur d'authentification si l'IP n'est pas belge. Le token serait le GSM ou smartphone des utilisateurs. Vu que tout GSM possède la fonctionnalité SMS, cette méthode peut donc s'appliquer pour tout le monde[4]. Grâce à ceci, les trois besoins peuvent être rencontrés. La méthode est peu coûteuse car le GSM est acheté par les utilisateurs. Le canal SMS est connu de tous, entrainant une utilisabilité correcte. Et la sécurité est renforcée grâce au second facteur d'authentification.
Le deuxième parcours de l'arbre a donné le résultat espéré.

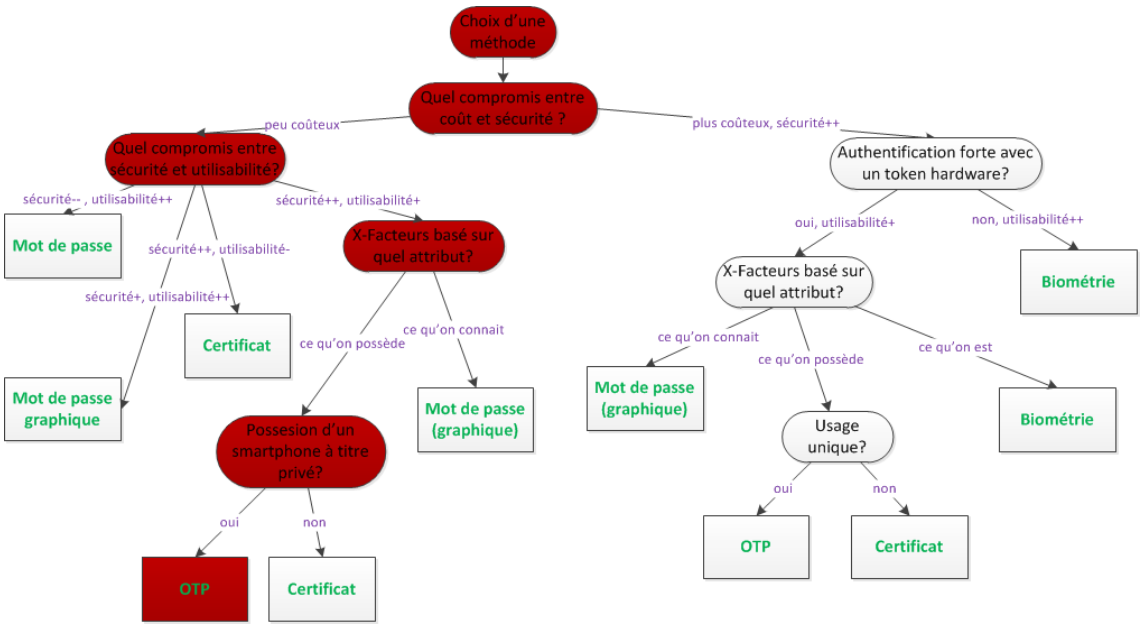

FIGURE 4.4 – Choix d'une méthode en fonction du dernier critère pour le cas UNamur

4. Nous partons de l'hypothèse que tout le monde possède son propre GSM.

Le responsable IT a néanmoins émis plusieurs remarques :

- Pour lui, la question en haut de l'arbre ne devrait pas être "quel compromis entre coût et sécurité ?" mais bien "**quel est le niveau de sécurité voulu ?**

- Il a aussi fait remarquer que l'authentification par certificat n'avait pas une utilisabilité faible mais au contraire, une forte. Si tout fonctionne bien, l'authentification par certificat est transparente pour l'utilisateur.

- La branche de droite représente une sécurité élevée donc le coût sera forcément élevé. La branche de gauche possède une sécurité plus faible par son moindre coût.

2. Dans le cas de l'arbre des protocoles, il était intéressé par une aide au niveau du choix de SSO. L'Unamur envisage d'implémenter un SSO avec des applications hétérogènes. Actuellemnt, un SSO de type CAS est en place pour les applications Java. Mais il trouve celui-ci limité. L'aspect authentification forte avec des protocoles n'était pas à envisager pour l'instant.

   Les besoins démandés concernant le SSO sont les suivants :

   (a) Le SSO ne doit pas se limiter aux applications écrites en Java.

   (b) Le SSO doit pouvoir être utilisé en interne et en externe.

   (c) Le SSO doit avoir une interopabilité correcte.

   Sachant qu'il est question d'un environnement ouvert et fermé, deux parcours de l'arbre sont nécessaires.

   Dans le cas de l'environnement fermé, l'idée du responsable était une solution basée sur SAML, comme c'est le cas pour la fédération Belnet des universités. Les décisions prises à travers l'arbre lui donne la même solution. Cependant, OpenID reste aussi envisageable car il possède les mêmes fonctionnalités de base que SAML.

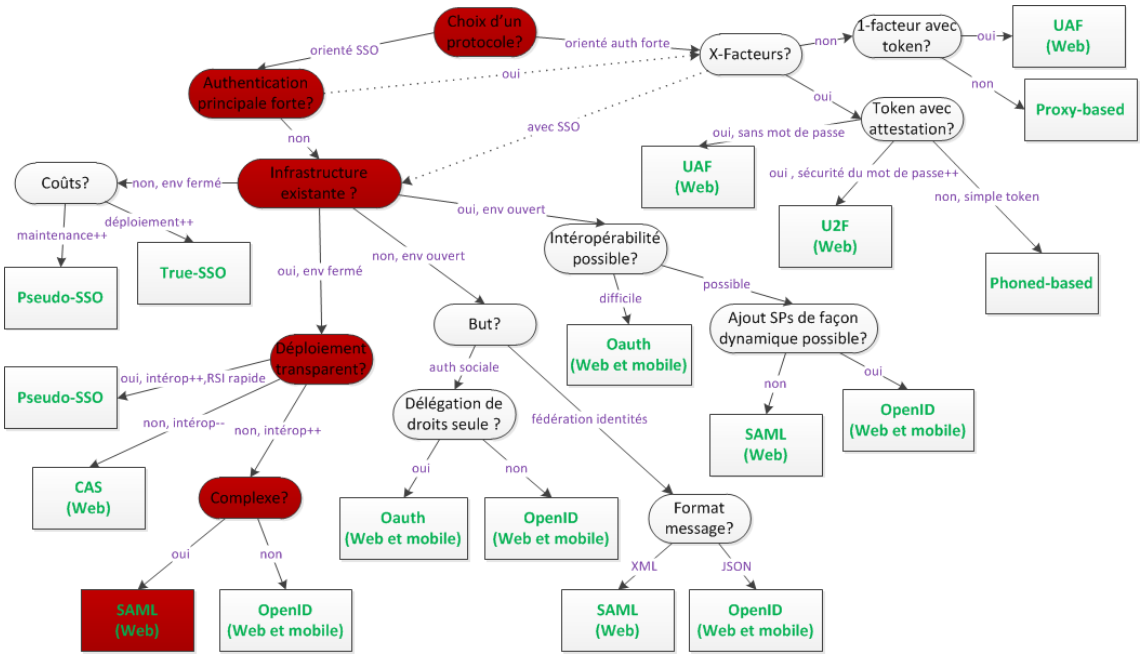

FIGURE 4.5 – Choix d'un protocole dans un environnement fermé pour le cas UNamur

Le second parcours à travers cet arbre lui donne la même solution aussi. Il a noté que OpenID avait

la possibilité d'ajouter dynamiquement les SPs par rapport à SAML et il trouvait cela intéressant lorsque le nombre de SPs devient conséquent.

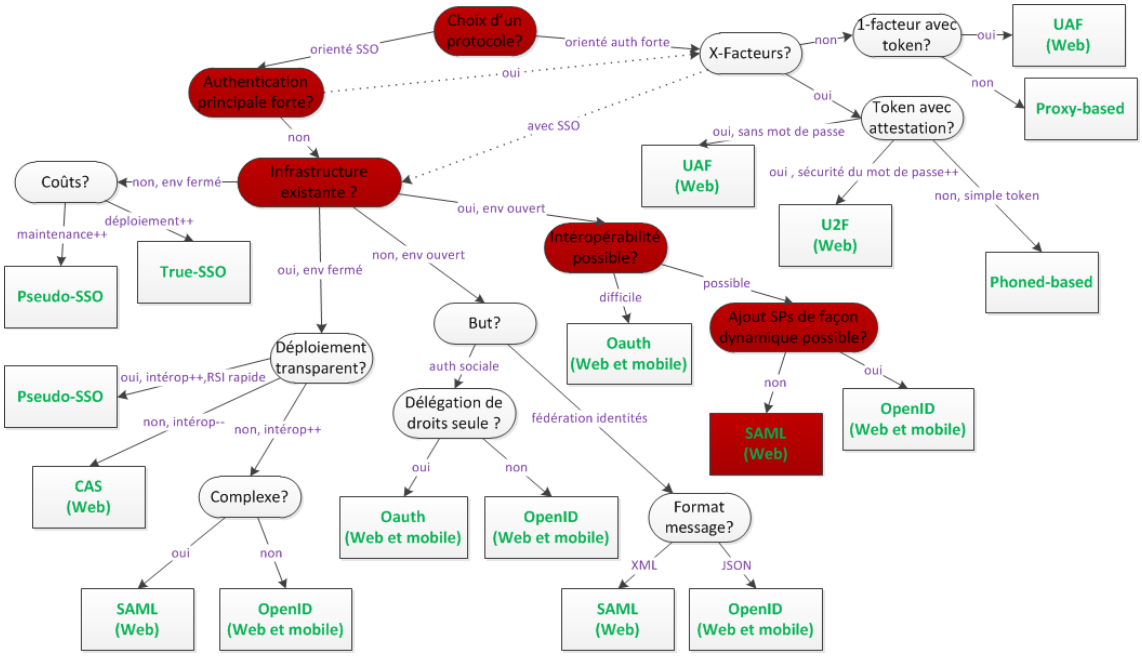

FIGURE 4.6 – Choix d'un protocole dans un environnement ouvert pour le cas UNamur

Il a emis deux remarques principales par rapport à l'arbre.

- Même s'il n'était pas intéressé par l'authentification forte lors du SSO, il aurait trouvé intéressant d'avoir un lien entre le choix final et la possibilité d'avoir une authentification principale forte lors du SSO. Mais il estimait que cela aurait probablement surchargé le schéma.
- Il aurait aimer voir un critère 'applications hétérogènes' dans l'arbre. Cependant, il a évoqué que l'interopabilité pouvait remplacer ce critère.

4.2.2 Cas II : UCM

Le second cas est notre milieu professionnel. Nous avons présenté les deux arbres à notre responsable informatique, qui s'est prêté à l'exercice.

1. Le cas d'utilisation dans lequel l'arbre des méthodes s'applique est une application dans un environnement ouvert, à partir de laquelle les clients encodent les prestations de leur ouvriers et employés. Il a évoqué deux besoins indispensables :

   (a) Les clients ne peuvent accéder qu'à leur seul profil et en aucun cas à celui des autres clients.

   (b) L'utilisabilité de la méthode d'authentification ne doit en aucun cas être un frein à l'utilisation de l'application.

   Le choix effectué via le parcours de l'arbre est le mot de passe classique. Il reflète bien l'implémentation existante de l'application et est en adéquation avec les besoins voulus.

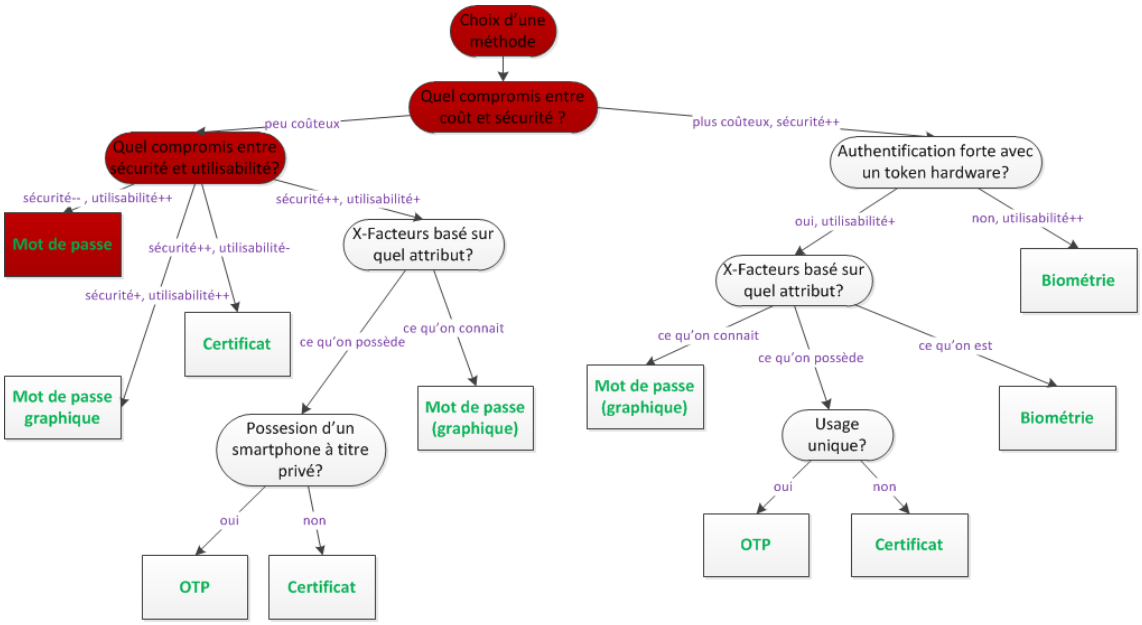

FIGURE 4.7 – Choix d'une méthode pour le cas UCM

Tout comme le responsable IT de l'Unamur, le notre a quand même évoqué certaines remarques vis à vis de cet arbre :

- Selon lui, la question de base ne devrait pas être "quel compromis entre coût et sécurité ?" mais bien "quel compromis entre **utilisabilité** et **sécurité** ?". Sachant qu'il y a un aspect marketing derrière l'application, l'utilisabilité est aussi importante que la sécurité.

De son opinion, le coût ne doit être évalué qu'après ces deux critères. Dès lors, il a dit qu'il choisirait intuitivement la branche de droite pour avoir une sécurité forte. Or, le choix final après lecture complète de l'arbre se situe dans celle de gauche.

- Il a trouvé que les sous-branches de gauche et de droite étaient fort ressemblantes. De ce fait, il se demandait pourquoi il n'y avait pas de méthode d'authentification par biométrie dans la branche de gauche sachant qu'avec les nouvelles technologies de smartphone, il était possible d'avoir une authentification biométrique à moindre coût.

- Il a trouvé que la branche de droite était assez figée par rapport à celle de gauche qui offrait plus de flexibilité. Il a ajouté qu'il aurait été mieux d'indiquer 'sécurité+++' pour la branche de droite, en jouant sur le fait qu'un système hautement sécurisé implique forcément un coût conséquent. Par contre, pour la branche de droite, il aurait fallu mettre 'sécurité++' en jouant avec le compromis utilisabilité versus sécurité.

2. Il n'était pas vraiment intéressé par les protocoles d'authentification forte. Cette branche a donc directement été écartée.

   Par contre, sachant que l'UCM a récemment changé son système de SSO, la branche concernant ce protocole a été utilisée afin de valider le choix effectué. Le SSO mis en place devait répondre a trois besoins :

   (a) Il devait s'intégrer dans un environnement hétérogène. Cela veut dire qu'il peut faire du SSO avec des applications tournant sous Tomcat, Jboss, Websphere ou encore des applications

propriétaires comme des CRM ou ERP.

(b) Il devait pouvoir être utilisé dans un environnement ouvert et fermé.

(c) Il devait si possible être transparent pour les applications.

Afin de couvrir le cas de l'environnement ouvert et fermé, nous avons du parcourir l'arbre deux fois. Dans le cas de l'environnement fermé, nous sommes bien arrivés au même résultat, c'est à dire Pseudo-SSO. Ce protocole SSO permet l'interopérabilité voulue dans un environnement fermé et hétérogène, et se déploie de façon transparente. L'implémentation faite par l'UCM est la solution Pseudo-SSO d'Evidian.

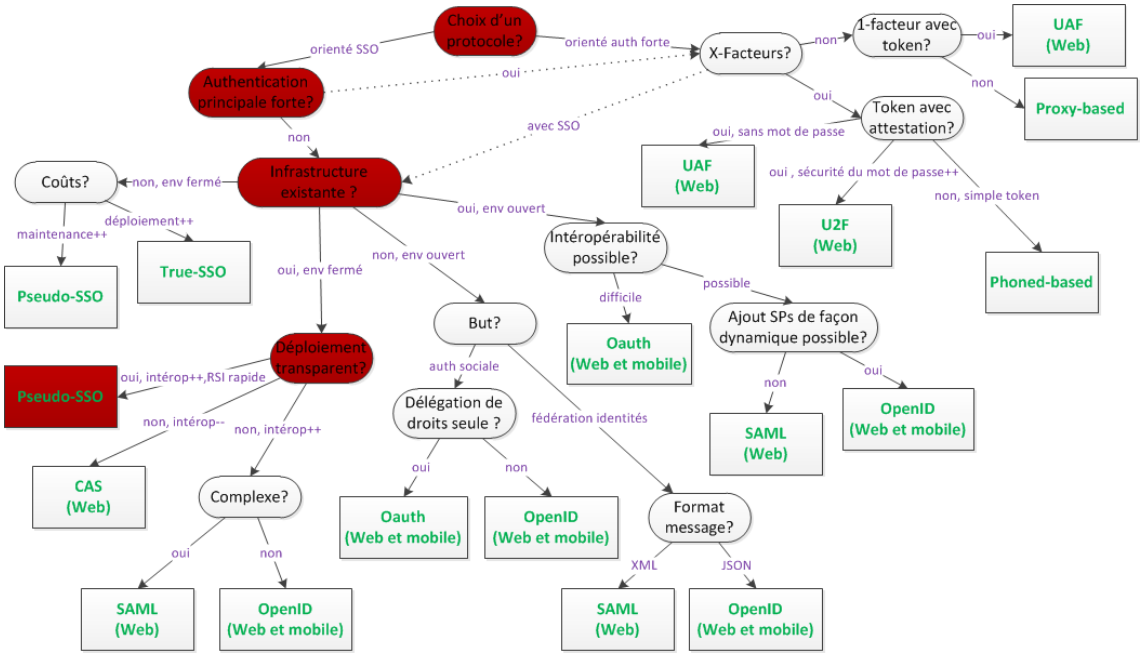

FIGURE 4.8 – Choix d'un protocole dans un environnement fermé pour le cas UCM

Par contre, dans le cas de l'environnement ouvert, nous ne sommes pas arrivés au même résultat. En suivant l'arbre, nous arrivons dans une solution de type True-SSO comme SAML ou OpenID. Or ce n'est pas le cas de l'implémentation, qui est la même que celle dans l'environnement fermé. Cela veut dire que notre arbre est potentiellement incomplet. Il devrait proposer une sous-branche vers les Pseudo-SSO.

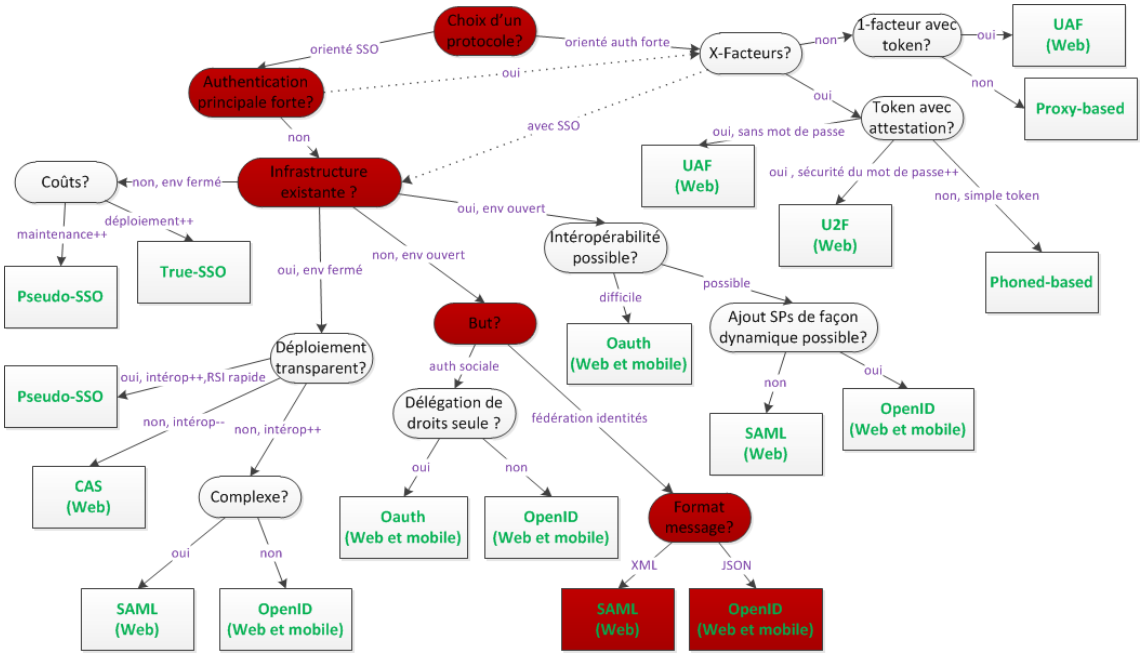

FIGURE 4.9 – Choix d'un protocole dans un environnement ouvert pour le cas UCM

Par rapport à ces deux parcours d'arbre, il a aussi émis des remarques :

- Pourquoi le critère 'application hétérogène' n'est-il pas présent dans l'arbre ?

- Est-il vraiment indispensable d'avoir deux sous-branches pour l'environnement ouvert, sachant que le choix final arrive vers les mêmes protocoles, et ce que nous soyons en présence d'une infrastructure existante ou non ? Il estime qu'il y avait moyen de simplifier ceci.

- Pourquoi les Pseudo-SSO ne sont-ils pas envisagés dans un environnement ouvert ?

4.3 Conclusion et adaptation

D'après les remarques émises concernant l'arbre des méthodes, nous pensons effectivement que des améliorations sont possibles. Nous voyons que la question de départ n'est pas adéquate et porte à confusion. Demander quel niveau de sécurité est désiré serait plus parlant. Maintenant, nous trouvons qu'il est quand même difficile d'avoir une question globale qui couvre l'ensemble des cas d'utilisation.

Nous pensons aussi qu'indiquer 'sécurité+++' pour la branche de droite serait plus concret, comme l'a cité notre responsable. Cela marquerait mieux le coût élevé par rapport au niveau de sécurité très élevé.

A propos de la branche de gauche, il serait intéressant de la diviser en deux sous-branches. Une première avec 'sécurité–' allant directement vers les mots de passe classiques et la seconde qui ressemblerait à l'existante. Il aurait cependant fallu adapter certains critères conformément à ce qui a été dit. C'est un fait, l'authentification par biométrie aurait pu apparaitre dans la branche de gauche dans le cadre de la possession d'un smartphone. Néanmoins, de notre opinion, ceci reste limité car la majorité des utilisateurs ne possèdent pas de smartphone pouvant réaliser une authentification biométrique.

Concernant la remarque à propos de l'utilisabilité des certificats, nous sommes d'accord que mettre 'utilisabilité-' est un peu excessif. Le critère 'utilisabilité+' serait plus adéquat car quand tout fonctionne

bien, cette authentification est très simple pour l'utilisateur. Elle devient plus compliquée en cas d'erreur. En tenant compte de tout ceci et en effectuant quelques améliorations, nous avons créé un nouvel arbre :

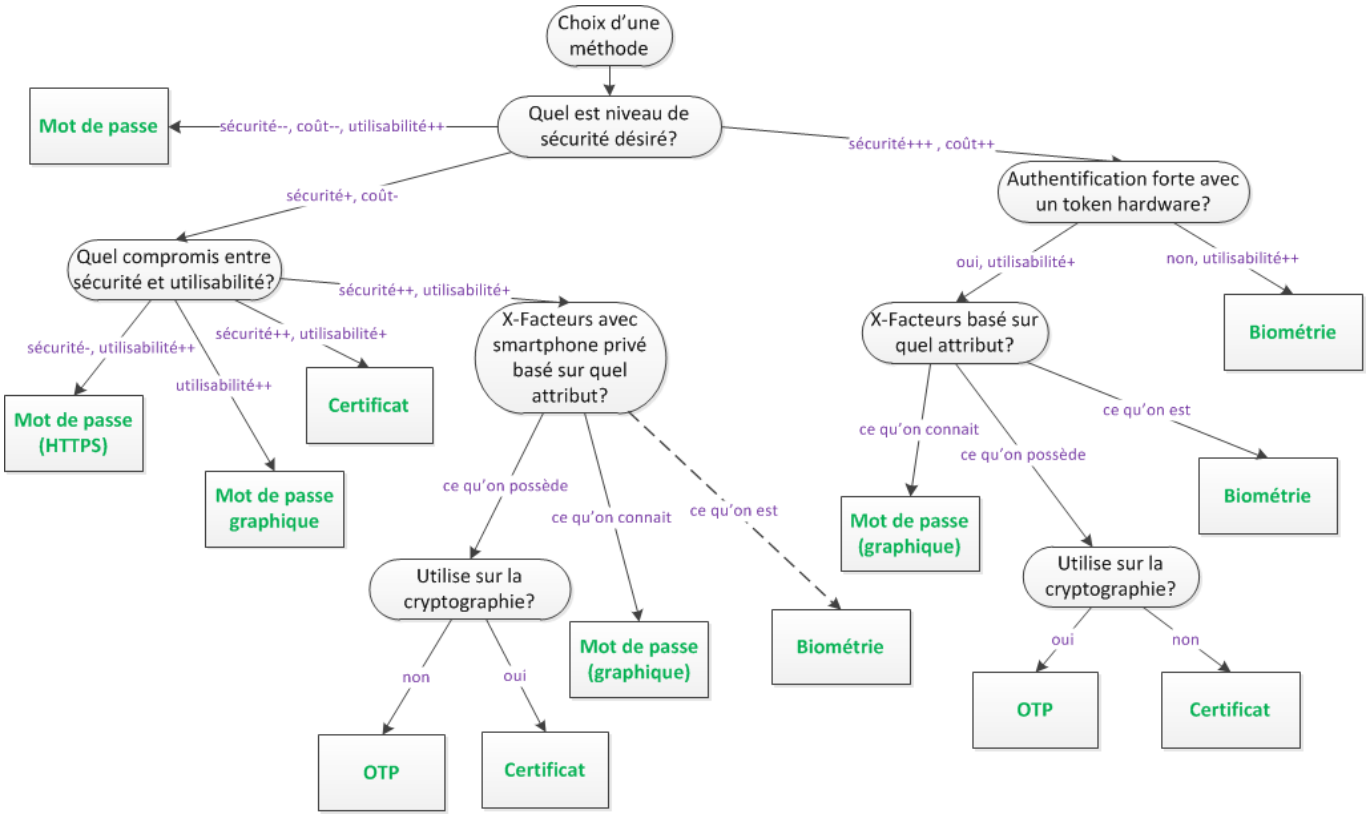

FIGURE 4.10 – Arbre des méthodes amélioré

Premièrement, nous avons modifié la question de base, en tenant compte de la remarque du cas Unamur. En second lieu, nous avons ajouté une branche supplémentaire à gauche allant directement vers les mots de passe en montrant une implémentation de la méthode qui soit facile à utiliser, peu chère mais peu sécurisée. A côté de ceci, nous avons adapté la branche existante :

- La branche propose une sécurité un peu plus forte que les mots de passe classiques avec un coût égal. Une exception faite pour une de ces sous-branches allant vers les mots de passe en HTTPS, ayant une sécurité- au lieu de –.
- Nous avons adapté l'utilisabilité des certificats, comme décrit ci-dessus.
- Nous avons regroupé le critère 'multi-facteurs' avec celui du 'smartphone privé' en proposant les trois alternatives classiques. Afin de distinguer les OTP des certificats, nous interpellons l'utilisateur sur une sécurité basée sur la cryptographie, que le certificat propose.
- Nous avons ajouté le facteur biométrie en pointant l'attention sur le fait que tous les smartphones ne possèdent pas cette technologie. Nous avons montré cette possibilité via des pointillés.

Si nous revenons au niveau de la première question et adaptant le critère vis à vis de la remarque émise, la branche de droite propose dorénavant une sécurité+++ avec un coût++. Cela veut dire qu'un système très sécurisé coûtera implicitement plus cher. Tout comme la branche de gauche, nous distinguons de la même façon OTP et certificats dans le cadre d'un multi-facteurs.

Les points d'attention soulevés au sujet de l'arbre des protocoles montrent qu'il manque certains choix et critères dans l'arbre. Typiquement, les Pseudo-SSO peuvent être utilisés dans tout contexte où le SSO est de mise. Même s'il n'est pas recommandé de les utiliser dans un environnement ouvert, dû au fait qu'ils ne gèrent pas les identités et qu'ils rejouent l'authentification, posant donc certains problèmes de sécurité, ils pourraient quand même apparaitre, moyennant un point d'attention.
Il est aussi indéniable que nous ne prenons pas en compte les applications hétérogènes mais nous les avons exclues par choix lors de la désignation des critères comme indiqué dans la section 4.1. Cependant, en fonction du niveau d'interopérabilité du protocole, nous pouvons dire que le critère est implicitement présent. Si le système est fortement interopérable alors il prendra en compte les applications hétérogènes.
Tout comme pour l'arbre précédent, nous avons tenté d'améliorer l'arbre sur base des remarques émises :

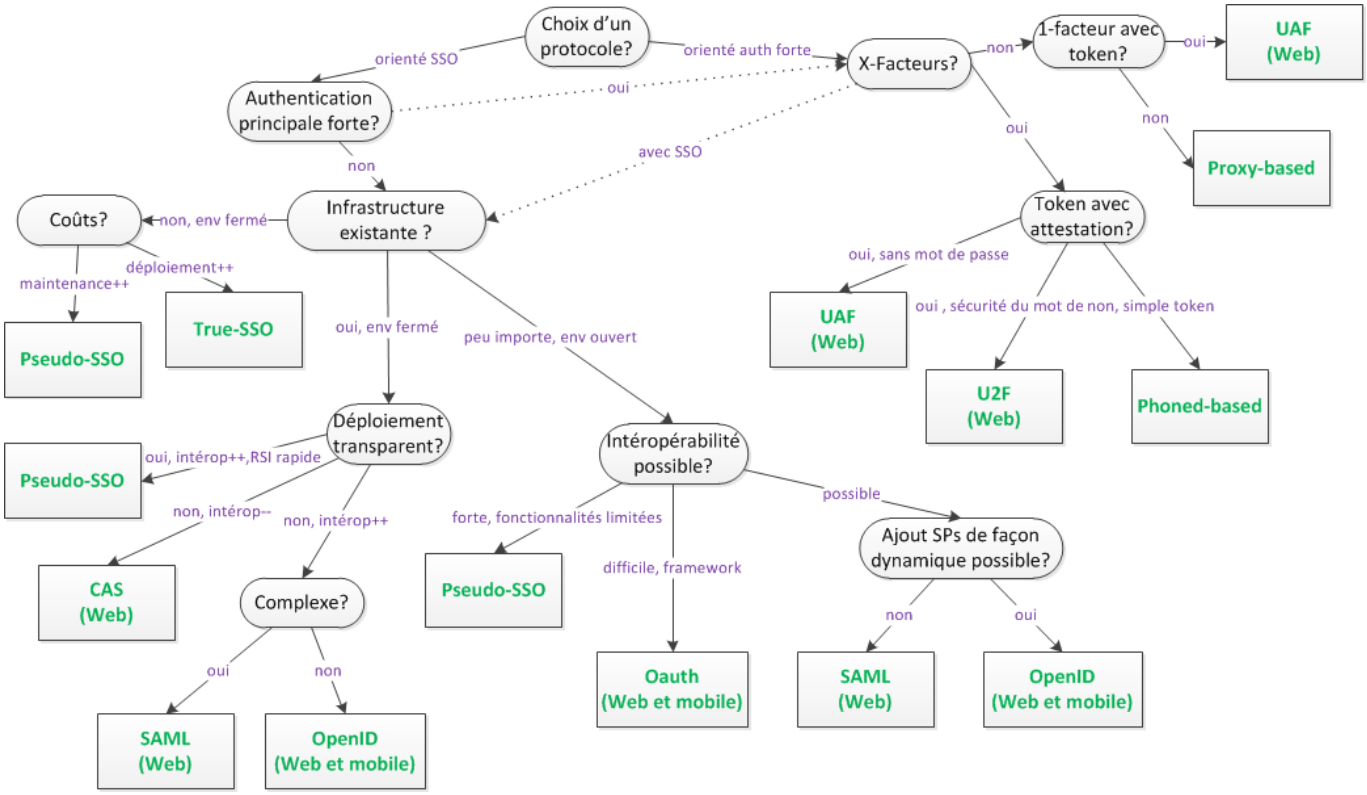

FIGURE 4.11 – Arbre des protocoles amélioré

Nous considérons que l'hétérogènité est gérée via l'interopérabilité des protocoles. S'ils sont interopérables, ils prennent nécessairement en compte ce critère. A ce niveau, aucun changement sera effectué dans l'arbre.
Comme l'a fait remarquer notre responsable, nous pouvons simplifier la branche concernant l'environnement ouvert. Sachant que le résultat se dirige à chaque fois vers les mêmes protocoles, peu importe que l'infrastructure soit existante ou non, nous avons regroupé les choix en une seule branche. Dorénavant, le choix sera effectué par rapport au critère d'interopabilité.
Nous avons ajouté une sous-branche avec les Pseudo-SSO en émettant le fait que les possibilités sont limitées dans le cadre d'un environnement ouvert.

Pour conclure, nous pensions que le critère le plus important pour un responsable IT était le coût mais nous avons vu que ce n'est pas spécialement le cas. Nous avons appris que l'architecte IT créé des infrastructures en terme de sécurité et d'utilisabilité puis les soumet aux clients [5] avec un prix. C'est à ce moment là qu'ils évalueront les possibilités existantes et qu'ils feront un compromis entre le coût et les autres critères par rapport aux fonctionnalités demandées.
Nous avons utilisé la première itération des arbres pour qu'ils soient critiqués par des personnes du métier. Dû à leur expérience, ils ont émis rapidement des remarques pertinentes dans le but d'une amélioration possible. Sur base de celles-ci, nous avons effectué une deuxième itération, qui nous semble déjà plus claire. Nous sommes conscients qu'ils pourraient certainement encore y avoir des améliorations mais il faudrait probablement avoir des avis supplémentaires pour peaufiner ces arbres.

5. Nous parlons ici de responsables IT, directeurs ou personnes du métiers.

CHAPITRE 5

Evolution et conscientisation

Nous avons vu tout au long de ce document que le remplacement des mots de passe était difficile. Il est possible de renforcer la sécurité via différentes politiques ou canaux sécurisés mais la problématique reste toujours d'actualité. Nous remarquons quand même une certaine évolution dans les habitudes des utilisateurs. Ils sont de plus en plus intéressés par des solutions plus sécurisées car ils veulent que les données restent sécurisées.
Ces dernières années, les attaques sur les mots de passe ont permis de dérober des millions d'informations au travers le monde. Les récents événements de vol de données personnelles ont permis de conscientiser une partie des utilisateurs. C'est aussi le cas des sociétés publiques et privées, qui pointent de plus en plus leur attention sur la sécurisation des données de leurs clients.

A travers ce chapitre, nous parlerons de plusieurs sujets. Dans un premier temps, nous étudirons la protection des données des utilisateurs. En second lieu, nous ferons l'état des lieux de l'évolution de l'authentification dans les services publiques. Un paragraphe sur les mobile devices tels que les smartphones et tablettes sera aussi débattu. Nous verrons qu'il existe des solutions prometteuses qui pourraient apporter une résolution aux différents problèmes posés en début de ce document. Pour terminer, nous tenterons d'apporter une réponse à certaines questions posées dans ce début mémoire.

5.1 Privacy

Les méthodes d'authentification que nous avons présentées tout au long de ce document nous ont montré que les SPs et IdP pouvaient collecter des données relatives aux utilisateurs. Certaines de celles-ci sont personnelles, et par conséquent critiques. Les données comme les numéros de carte bancaire, les données de carte visa ou encore celles présentes dans les puces des cartes d'identité doivent être préservées et ne peuvent être disperçées sur le Web. Ce fait ne se pose pas seulement que pour le Web mais aussi en pour les systèmes d'authentification internes dans les entreprises. Un rapport de la SANS Institude[SANS-Institute, 2002] a fait remarquer qu'il s'avérait que la plupart des utilisateurs impliqués dans un système utilisant la biométrie, se sentaient concernés ou étaient inquiets quant à l'utilisation de leurs données présentes dans le système.

Nous savons que les réseaux sociaux tel que Facebook, Twitter , Linked'In ou encore MySpace [1],utilisent ces données lors de la phase d'authentification et afin de définir les autorisations adéquates. Néanmoins ils les utilisent aussi dans un cadre commercial, voire juridique.
Par exemple, les conditions générales de Facebook[Facebook, 2016] stipulent que n'importe qui peut utiliser le contenu présent sur le mur des utilisateurs. Dès lors, certaines agences ne se privent pas de le faire. C'est notamment le cas de celles qui travaillent dans le monde de la mode, qui prennent des photos des utilisateurs totalement à leur insu et bien sur, sans leur consentement. L'utilisateur peut partiellement contrer ceci en modifiant ses paramètres de confidentialité de son profil. Nous disons partiellement, parce que si un de ses amis publie une photo de lui sur son mur et que son profil est ouvert à tout le monde, alors les paramètres de confidentialité du premier individu tombent à l'eau. Il faut que chacun d'entre eux aient les mêmes paramètres.
Si dans la théorie, ceci est envisageable, ça ne l'est pas du tout en pratique. Du moins, cela se voit très peu. Tout ceci pour dire que la sécurisation des comptes utilisateurs et de leurs données ne dépend pas entièrement des sociétés mais aussi des utilisateurs eux-mêmes. Il est souvent dit que "la sécurité des données est égale à la sécurité du maillon le plus faible", et dans ce cas-ci, c'est l'utilisateur final.

Nous avons vu qu'il existe des contre-exemples aux réseaux sociaux. C'est notamment le cas du framework CEF eID. Les chapitres précédents ont montré que ce dernier utilise les données privées des utilisateurs dans un but d'authentification ou d'autorisation et à des fins purement non-lucratives. Lors du développement de ce framework, la Commission Européenne a mis un point d'attention sur le respect de la vie privée des utilisateurs sur le Web. Elle s'est efforcée de suivre des règles ou directives légales pré-établies. Lors d'une authentification à travers les frontières, le consentement du transfert de ses données vers un pays voisin sera explicitement demandé à l'utilisateur. Il pourra ainsi voir quelles données seront transférées vers quel(s) SP(s) et dans quel but.

Cet exemple permet d'introduire la problématique de la protection des données privées lors des échanges entre SPs et IdPs. Lors d'un flux SSO, et particulièrement les True-SSO, les IdPs envoient des flux d'informations comprenant généralement le statut de l'authentification, et des informations concernant l'utilisateur en fonction des besoins du SP. Cependant, le besoin d'informations des SPs est parfois un peu vague. Il arrive très souvent qu'il y ait un surplus d'informations transférées. Or, dans la plupart des cas, elles ne sont pas utiles pour le business du demandeur. Il est donc pertinent de se poser la question de la destination finale de ces données, de leur durée de vie, ainsi que de la juridiction autour de celle-ci.

- Est-ce que l'utilisateur a le contrôle sur ces échanges ?
- De plus, que se passe-t-il en cas de compromission, d'un IdP ou d'un SP malicieux ?

Nous savons qu'il existe des lois relatives au respect de la vie privée mais nous remarquons qu'il persiste un certain flou autour de celles-ci. De plus, une loi au sein des membres de l'UE n'est pas spécialement

1. Cette liste n'est bien sur pas exhaustive.

applicable en Amérique ou en Asie.

5.1.1 Les règles et directives existantes

Depuis la sortie et l'expansion du Web, l'OCDE et la Commission Européenne ont publié un ensemble de directives par rapport à la politique sur la vie privée. En 1980, l'OCDE a écrit ces lignes directrices régissant la protection de la vie privée et les flux transfrontières de données à caractère personnel[OCDE, 2016]. C'est ainsi que certaines directives importantes comme la 95/46/CE[Européenne, 1995] ont été créées. Cette dernière est importante car elle impose que les données personnelles collectées par des sociétés ne puissent être utilisées sans l'accord explicite de l'utilisateur auquel elles appartiennent[Wikipedia, 2016b]. Cette dernière s'applique à tous les membres de l'UE. Son but est de pouvoir faciliter la libre circulation des informations privées de manière sécurisée et contrôlée. Dès lors, tout utilisateur a le droit de demander les données le concernant à la société qui les possède. Il a aussi la possibilité de la modifier ou de les supprimer. Différents principes ont été mis en place au niveau national :

- "Principe de la limitation en matière de collecte : des limites à collecte doivent être établies, avec le consentement de l'utilisateur."
- "Principe de la qualité des données : les données doivent être pertinentes par rapport aux besoins de la société qui les récolte. La finalité concernant celles-ci doit être établies avant la fin de la collecte."
- "Principe de la limitation de l'utilisation : l'utilisation des données doit être strictement faite dans le cadre des activités de la société, sous l'accord de leur propriétaire."
- "Principe des garanties de sécurité : la garantie de la sécurité et de la transparence des données doit être établie."
- "Principe de la participation individuelle : toute personne peut demander l'accès à ses données personnelles dans un labs de temps raisonnable."
- "Principe de la responsabilité : toute société qui détient des données privées est responsable par rapport aux principes énoncés ci-dessus."

Il existe bien sur d'autres principes au niveau international [2]. Nous allons les citer tels qu'ils sont présentés dans l'article[OCDE, 2016]

- "Les pays Membres devraient prendre en considération les conséquences pour d'autres pays Membres d'un traitement effectué sur leur propre territoire et de la réexportation des données de caractère personnel."
- "Les pays Membres devraient prendre toutes les mesures raisonnables et appropriées pour assurer que les flux transfrontières de données de caractère personnel, et notamment le transit par un pays Membre, aient lieu sans interruption et en toute sécurité."
- "Un pays Membre devrait s'abstenir de limiter les flux transfrontières de données de caractère personnel entre son territoire et celui d'un autre pays Membre, sauf lorsqu'un ce dernier ne se conforme pas encore pour l'essentiel aux présentes Lignes directrices ou lorsque la réexportation

2. Nous parlons ici au niveau de l'UE.

desdites données permettrait de contourner sa législation interne sur la protection de la vie privée et des libertés individuelles. Un pays Membre peut également imposer des restrictions à l'égard de certaines catégories de données de caractère personnel pour lesquelles sa législation interne sur la protection de la vie privée et les libertés individuelles prévoit des réglementations spécifiques en raison de la nature de ces données et pour lesquelles l'autre pays Membre ne prévoit pas de protection équivalente."

- "Les pays Membres devraient éviter d'élaborer des lois, des politiques et des procédures, qui, sous couvert de la protection de la vie privée et des libertés individuelles, créeraient des obstacles à la circulation transfrontière des données de caractère personnel qui iraient au-delà des exigences propres à cette protection."
- "Les pays Membres devraient, sur demande, faire connaître à d'autres pays Membres les modalités détaillées de l'application des principes énoncés dans les présentes lignes directrices. Les pays Membres devraient également veiller à ce que les procédures applicables aux flux transfrontières de données de caractère personnel, ainsi qu'à la protection de la vie privée des libertés individuelles, soient simples et compatibles avec celles des autres pays Membres qui se confirment aux présentes lignes directrices."
- "Les pays Membres devraient établir des procédures en vue de faciliter l'échange d'informations relatives aux présentes lignes directrices et l'assistance mutuelle lorsqu'il s'agit des questions de procédure et d'échange réciproque d'information."
- "Les pays Membres devraient s'employer à établir des principes, au plan intérieur et international, afin de déterminer le droit applicable en cas de flux transfrontières de données de caractère personnel."

Les principes cités précédemment prouvent qu'il existe des textes à propos de l'utilisation des données cependant tout ceci était encore assez flou dû au fait que les législations entre pays membres étaient différentes. C'est pour cette raison qu'en 2012, la Commission Européenne a créé un projet permettant aux utilisateurs d'avoir une sécurisation de données plus accrue ainsi que plus de flexibilité par rapport à la gestion de celles-ci[Européenne, 2012].
Les sociétés publiques et privées sont désormais soumises à des conditions plus strictes si elles veulent collecter ces données. Une directive importante est que chaque société qui collecte des données privées au sein de l'UE se doit de les protéger. En cas de compromission, l'utilisateur pourra se retourner contre la société. De plus, dans chaque pays, il existe des organismes comme la Commission nationale de l'informatique et des libertés [3], qui agissent en tant que contrôleur des données informatiques privées qui sont stockées par les sociétés. Ces organismes se basent sur la loi de 1978 relative à l'informatique afin de faire respecter l'ensemble de ces directives.

3. Organisme spécifique à la France

5.2 Services publics

La problématique de la sécurisation des données ainsi que le respect de la vie privée des utilisateurs devient de plus en plus prise en compte par l'ensemble des entreprises. Le secteur privé est depuis longtemps sur le front. Les multinationales ont mis en place des méthodes d'authentification plus sécurisées depuis plusieurs années. Le secteur public reste quant à lui un peu à la traine. Ceci est explicable par le fait que le but de ceux-ci n'est pas toujours financier. Dès lors, l'aspect client est parfois laissé de côté, contrairement aux entreprises privatisées. De plus, l'étude, l'expertise et la mise en place de ces méthodes nécessitent un budget plus conséquent que le déploiement de simples mots de passe.

La plupart des infrastructures informatiques concentre leur authentification en interne, de façon centralisée. Certaines d'entre elles sont lourdes en terme de maintenance et de sécurisation. Il est donc parfois plus avantageux de déléguer cette tâche à des organismes dédiés. De ce fait, l'authentification devient complètement décentralisée, avec les avantages et incovénients que celà comporte.
En Belgique, l'organisme CSAM[CSAM, 2015] est un système qui permet aux administrations de décentraliser leur authentification. Il propose trois méthodes d'authentification différentes selon le niveau de sécurité requis : le mot de passe simple, le certificat de la carte eID ou un mot de passe, accompagné d'un token hardware avec code.
Le principe de celui-ci est d'authentifier l'utilisateur et de lui donner accès à des services publics en fonction des attributs présents sur sa carte eID. L'age, le sexe, la commune sont les plus utilisés lors de cette phase. Néanmoins, la phase d'autorisation n'est pas obligatoirement gérée par CSAM et sa gestion peut être réalisée au niveau de l'administration publique elle-même.
CSAM permet cette gestion du citoyen en ligne par le biais du protocole SAML. Il propose trois grandes fonctionnalités :

- FAS[Fedict, 2014] permet l'authentification et l'identification des personnes physiques.
- GGA permet aux employés d'une entreprise d'accéder aux services de l'état, moyennant un enregistrement de celle-ci.
- Gestion des mandat : "chaque type de mandat donne accès à une application électronique du secteur public. Le mandataire peut réaliser les actions mentionnées dans le mandat, après qu'il ait été approuvé 'pour accord' par les deux parties."

Les institutions publiques comme les communes, les guichets ou les administrations gouvernementales proposent de plus en plus de services en ligne aux citoyens. L'e-Gouvernement est devenu omniprésent sur du Web. Dès lors, elles commencent à se diriger vers des solutions comme CSAM. En décentralisant leur authentification, elles gagnent du temps et de l'argent tout en ayant une sécurité et une protection des données privées accrues. De plus, dû à la technologie SAML et à son profil Web Browser SSO, CSAM permet un SSO entre services publics, ce qui n'est pas pour déplaire aux utilisateurs.

Au niveau international, le même type de framework existe. Nous l'avons décrit dans les chapitres

précédents, il s'agit de CEF-eID. Tout comme CSAM, il permet un accès transfrontalier sécurisé. De plus, il peut autant s'appliquer aux services publics qu'aux privés. Sa mise en place requiert du temps, de l'argent et un niveau de compétence élévé. Cependant, la Commission Européenne propose un suivi et des formations adéquates.
Par rapport à CSAM, qui est déjà en place depuis quelques années et qui est donc mature, CEF-eID est encore peu mature du fait qu'il soit peu déployé sur le Web. Cette expansion pourrait renforcer les échanges de flux sécurisés entre pays, ce qui serait un avantage considérable pour les membres de l'UE.

5.3 Mobile devices

Le Web est en plein croissance et l'évolution des habitudes des utilisateurs va de pair avec celle-ci. De nos jours, tout le monde est connecté en permanence grâce aux tablettes et smartphones. Les mobile devices ne sont plus utilisés que pour surfer ou discuter avec des amis mais aussi pour lire ses emails, acheter des articles en ligne ou encore effectuer des transactions bancaires. Ceci en fait des cibles de choix pour les hackeurs. C'est le constat que la plupart des firmes de sécurité font ressortir. Le vol d'informations sensibles sur ces appareils est devenu monnaie courante.
En terme d'utilisabilité, nous remarquons que ces devices n'appréhendent pas l'authentification de la même façon qu'une machine classique. Le clavier de ces derniers est beaucoup plus petit et les utilisateurs ne sont très enclin à taper un mot de passe ou un code avec. Très souvent, ils tentent de contourner ceux-ci soit en les remplaçant, soit en les supprimant. De ce fait, la sécurité du système est parfois mise en péril au profit de l'utilisabilité.

Les nouveaux modèles de smartphones proposent d'ailleurs de ne plus utiliser les codes ou les mots de passe pour accéder à l'interface principale mais bien d'autres méthodes plus sécurisées. Si pour les experts en la matière, le système doit être plus sécurisé, l'utilisateur, quant à lui veut un système facile à utiliser. De ce fait, les méthodes comme l'authentification graphique ou la biométrie sont devenues beaucoup plus répandues qu'auparavant.
La tendance à utiliser la biométrie comme authentification sur mobile device se montre d'ailleurs très positive. En effet, elle permet offrir aux utilisateurs un très bon compromis entre utilisabilité et sécurité malgré qu'elle soit parfois controversée par sa mise en place et son taux de faux positifs / négatifs.
Apple est une société qui mise beaucoup sur l'ergonomie et l'utilisateur de ses IPhones. Dans les versions 5 et 6 de ces smartphones, la fonctionnalité TouchID[Apple, 2016b] a été ajoutée. Cette dernière est très simple et permet de déverrouiller l'interface principale avec son doigt. Elle permet aussi d'accéder à différentes applications comme l'iTunes Stores, l'App Store ou encore l'iBooks Store.
Dans le cadre de transaction bancaire, Apple Pay[Apple, 2016a] est une des nouvelles solutions existantes. Elle permet d'effectuer des paiements électroniques sans qu'aucune carte bancaire ne soit utilisée. Lors d'une transaction comme un paiement bancaire au magasin, l'utilisateur présente son smartphone devant un lecteur et ensuite active le module biométrique avec son doigt. Par ce geste anodin, l'utilisateur sera authentifié. L'application va alors générer un ID unique de telle façon que le numéro de carte bancaire et

le code ne transitent pas sur le réseau. Cet ID réprésentera et liera l'utilisateur ainsi que ses données au niveau de l'organisme bancaire. De cette manière, il sera difficile de voler quelques informations concernant la transaction sur le smartphone.
La limite de ce système se situe au niveau du comportement humain. Un individu mal attentionné pourrait forcer une personne à s'authentifier contre sa volonté.

Il existe bien sur d'autres exemples comme les deux que nous venons de citer. Même si ces exemples sortent du cadre Web vers lequel ce mémoire est axé, nous pensons qu'il est important de faire une petite parenthèse pour montrer cette évolution. Les smartphones permettent d'étendre l'authentification biométrique de manière plus aisée. Sachant qu'elle est plus sécurisée que les mots de passe, nous pensons qu'il serait intéressant que ce système soit appliqué à l'Internet Mobile. En poussant les développeurs à créer une authentification comme telle, les mots de passe pourraient très bien voir leur existance arriver à leur terme.
De notre opinion, cette option est à envisager.

5.4 Futur de l'authentification

Les recherches en matière de sécurisation de l'authentification continuent d'avancer et de s'améliorer. L'intérêt des sociétés et des experts en sécurité se porte sur une solution en particulier : FIDO.
Dans l'état de l'art, nous avions décrit ce protocole. Bien qu'encore peu présent sur le Web [4], il commence à être déployé de plus en plus par de grandes entreprises. Conscientes des risques que comportent les mots de passe pour la pérénité de leur business, elles ont pris le choix de migrer leur système d'authentification vers ce framework sécurisé. Cette prise de décision a impliqué des changements au niveau de l'infrastructure IT mais aussi au niveau des utilisateurs.
Du point de vue de l'infrastructure, elle a du être mise à jour afin de créer une PKI dédiée à FIDO. De plus, les applications ont dû être adaptées pour pouvoir récupérer les clés publiques des machines clientes, lors de leur enregistrement. Tout ceci demande du temps, de l'expertise et bien sur de l'argent.
Les utilisateurs voient aussi les habitudes perturbées. L'utilisation d'une clé telle que la Yubikey peut perturber certains d'entre eux. Le fait d'utiliser UAF encore plus. Cela nécessite donc un changement des habitudes, voire une formation pour plusieurs d'entre eux. De plus, certains pourraient être rétissants par le fait de devoir porter un token sur eux. L'avantage de FIDO avec les tokens hardware est que même si la clé est perdue ou oubliée par l'utilisateur, celui-ci pourra utilisé une autre clé très rapidement. Contrairement à certains autres tokens, avec lesquels les procédures de renouvelement sont aussi lourdes.

Nous remarquons quand même dans la littérature que U2F est le framework le plus utilisé. Cette solution, couplée avec la clé Yubikey offre une sécurité et une utilisabilité optimales pour l'utilisateur final.
Plusieurs grandes multinationales comme Google, Facebook, Paypal ou encore la banque centrale d'Amé-

4. Environ 250 sociétés au travers le monde.

rique ont d'ailleurs choisi cette solution.
Récemment GitHub[Davenport, 2015] a fait de même. C'est un portail internet utilisé par les développeurs afin d'héberger et de partager du code Open Source publiquement mais aussi de manière privée. Il est comparable au gestionnaire de sources SVN, que nous avons utilisé dans le cadre du laboratoire de 1ère master.
Nous le savons, quand nous parlons d'aspect privé comme évoqué ci-dessus, cela va de pair avec un niveau de sécurité. En implémentant les spécifications FIDO, GitHub veut attirer l'attention sur le renforcement de l'authentification Web et encourager les développeurs à mettre en place ce système dans leur phase de développement. Via ce portail, très connu sur le Web, cette entreprise espère changer les habitudes et la vision des gens afin de pousser les mots de passe vers la sortie.

Nous le voyons, ce framework est une bonne nouvelle pour pallier aux problèmes de sécurité de l'authentification. En version 1.0, FIDO a créé les frameworks UAF et U2F. La version 2.0 se base sur les mêmes spécifications que 1.0 mais va plus loin.
En collaboration avec W3C-Charter[Halpin, 2016], un groupe de travail qui a été créé par le World Wide Web Consortium[5], FIDO veut créer une API sécurisée, utilisable par tous les navigateurs et toutes les plateformes Web. Pour ce faire, FIDO a partagé ses spécifications dans le but qu'elles soient intégrées dans les standards de développement que W3C publie. L'API qui en découlera, sera basée sur la cryptographie asymétrique et permettra la signature électronique ainsi que l'attestion des propriétés du signataire.
Le but de celle-ci est d'éviter toute utilisation de secret partagé comme les mots de passe et de rendre impossible toute attaque par phishing. De plus, elle a été prévue afin d'encourager l'utilisation des authentifications multi-facteurs et des tokens hardware. Cette API est un script comme par exemple un Javascript, qui sera utilisé à travers le navigateur et seulement avec l'accord de l'utilisateur.
Elle sera appelée pour les deux phases classiques que nous connaissons : l'enregistrement et l'authentification.
Afin de pouvoir les réaliser, les authenticators dont nous avions parlés dans l'état de l'art vont être utilisés. Il en existe deux types[Hubert Le Van Gong, 2015] :

- Internes : ils exécutent leurs opérations sur la même machine comme par exemple un smartphone ou encore un ordinateur classique. Ceci est généralement réalisé via une plateforme sécurisée dont le processus tourne sur cette même machine[6].
- Externes : les opérations se déroulent sur un device externe qui est accédé via USB, Bluetooth ou NFC.

L'enregistrement est très simple par rapport à d'autres systèmes. Si nous prenons l'exemple donné dans la documentation[Hubert Le Van Gong, 2015] où l'authentificator est interne et le device est un smartphone, la phase se fera en cinq étapes.

1. "L'utilisateur va surfer sur un site et effectuer une signature électronique en utilisant la méthode

5. Le terme W3C est plus souvent utilisé.
6. Les termes utilisés sont généralement Trusted Platform Module ou Secure Element.

voulue."

2. "Via un message, le smartphone va demander si le device doit être enregistré auprès du site."
3. "Le consentement de l'utilisateur est donné."
4. "Le téléphone demande à l'utilisateur une méthode d'autorisation comme par exemple un code PIN ou empreinte digitale."
5. "Une fois la méthode donnée, le site indiquera que l'enregistrement est terminé."

Comme nous pouvons le voir, cette étape est triviale pour l'utilisateur. La phase d'authentification va dans le même sens.
Pour décrire cette phase, la documentation prend l'exemple d'un smartphone agissant comme device (authenticator externe) mais l'utilisateur pourrait utiliser n'importe quel authenticator interne ou externe. En fonction du type de celui-ci, un message sera affiché sur le device ou non.

1. "Depuis sa machine, l'utilisateur va se rendre sur le site et indiquer qu'il veut signer via son smartphone."
2. "Un message lui sera affiché en lui demandant d'effectuer l'action d'authentification sur ce dernier, afin de finaliser celle-ci."
3. "Du côté du smartphone, un pop-up discret apparaitra lui proposant de continuer l'action."
4. "Une fois ceci fait, il sera invité à choisir un profil et fournir le geste d'autorisation (PIN, Biométrie ou autre)."
5. "Après cet ensemble d'actions, la page sera affichée dans navigateur avec l'utilisateur authentifié."

Le rôle de l'API dans ces deux phases est la génération de paires de clés et de l'identifiant lors de l'enregistrement ainsi que la vérification de la possession de la clé privée par la navigateur lors de l'authentification. La phase d'attestation dont nous avions parlé ci-dessus est réalisée par l'authenticator. Une attestion peut être délivrée au site Web pour chaque clé publique qui représente un identifiant d'un utilisateur. Cette attestation contient une signature réalisée par la clé d'attestation et est embarquée dans la clé publique.

Si le but de l'API est de faire une authentification sécurisée et utilisable pour tout un chacun, le but final de FIDO est de faire adopter ces standards de développements à l'ensemble des utilisateurs et faire en sorte que les mots de passe disparaissent complètement. Dans le futur, FIDO ne compte pas s'arrêter au Web. De par leur but, leurs spécification ne sont s'appliqueront plus seulement au Web mais à toute plateforme où l'authentification est possible (Desktop, WIFI, Bluetooth, Unix, Windows, etc...).

5.5 Réponse aux questions : Partie III

Dans cette troisième partie, nous répondrons aux questions restantes exposées dans l'état de l'art. Si nous revenons sur la question de l'opportunité de remplacer les mots de passe avec l'expansion des

mobiles devices, nous pouvons voir qu'il y a réellement une opportunité de changer les habitudes.
Vu que les utilisateurs évitent les mots de passe avec les mobiles devices pour les raisons citées ci-avant, les sites Web pourraient par exemple envisager un déploiement massif d'un système biométrique dédié à l'authentification pour smartphones et tablettes. Ils pourraient détecter l'appareil utilisé lors de la connexion et rediriger vers ce type d'authentification.

FIDO avec son API sécurisée, va aussi dans ce sens grâce ses authenticators. L'avantage de ceux-ci est qu'il n'y a pas de limitations de la méthode d'autorisation. Cela peut être un code PIN, un certificat, un OTP, de la biométrie, etc... Ce système permet aux entreprises de choisir leur politique d'authentification et de garder son authentification sécurisée.
Nous pensons que le fait d'impliquer de grands acteurs connus du Web tels que Google, GitHub, Paypal et W3C, est primordial pour que les habitudes changent. Les utilisateurs ne mettront pas en place des solutions qui n'ont pas fait leurs preuves.
L'intégration des spécifications FIDO dans les standards de développement de W3C pourrait réellement changer la donne. Cependant, de notre opinion, ceci est clairement orienté pour les entreprises qui les suivent. Or ce n'est pas le cas de beaucoup d'entre elles. Suivre des standards demande plus de rigueur et de temps, et donc implicitement plus d'argent. Pour les grandes entreprises, ce n'est pas réellement un problème parce que tout est bien cadré. Par contre, nous émettons certaines craintes par rapport aux autres sociétés, de taille moindre. Si la mise en place d'un framework tel que FIDO peut réellement rendre l'authentification sécurisée, sa complexité pourrait être un frein. Les PKI, les tokens hardware, la biométrie ne sont pas toujours simples à gérer et à déployer. Ceci nécessite une organisation et un suivi quotidien. De ce fait, les mots de passe seront encore choisis comme méthode standard pour ces entreprises ainsi que pour les utilisateurs lambdas.
Pour ces raisons, nous pensons que les mots de passe seront surement encore présents dans le futur. Mais leur présence sur le Web sera surement diminuée par les nouveaux outils comme FIDO. De manière générale, l'authentification sur le Web devient plus sécurisée qu'auparavant. Les techniques utilisées par les hackeurs sont plus complexes parce que les défenses le sont aussi. Les nouvelles brèches de l'authentification ainsi que leurs conséquences attirent beaucoup plus l'attention qu'auparavant. Nous voyons une conscientation globale des acteurs sur le Web.

CHAPITRE 6

Conclusion

Ce mémoire fut un travail de longue haleine et pour lequel nous nous sommes beaucoup impliqués. A travers celui-ci, nous avons tenté d'en savoir plus sur l'authentification, et plus précisement à propos de la problématique des mots de passe. Au départ de ce sujet, nous ne nous doutions pas que la sécurité des mots de passe était à ce point importante. Les recherches que nous avions menées nous ont permis de voir plus clair et de prendre du recul par rapport à la sécurité de nos propres mots de passe ainsi que celle des systèmes que nous gérons dans le cadre professionnel.

Afin d'étudier la problématique des mots de passe, nous nous sommes interrogés sur leur présence dans les systèmes, leur sécurité ainsi que leur évolution depuis leur création. Dans le but de cadrer le sujet, nous avons orienté notre reflexion vers l'authentification Web. Nous avons pu ainsi poser les grandes questions relatives à ce mémoire. Ces dernères concernaient l'évolution des mots de passe, de leur sécurité, leur présence sur le Web ainsi que les alternatives pour les remplacer ou les éliminer. Nous nous sommes aussi posés la question de la possibilité et la faisabilité d'un SSO global.
Pour y répondre, nous avons dû réaliser une étude de l'existant. Les différents articles scientifiques que nous avons lus nous ont permis de dresser un tableau général des méthodes d'authentification existantes avec leurs avantages, leurs inconvénients ainsi leur contexte d'utilisation. Nous avons ensuite décrit leur fonctionnement et ce qu'ils apportaient dans le contexte Web qui nous occupait. Nous avons prêté une attention particulière à distinguer les protocoles des méthodes d'authentification.
Cette partie nous a permis de dresser l'état de l'art sera lequel nous nous sommes basés pour la suite.

Dans la partie suivante, nous nous sommes intéressés au concept du SSO. Nous avons vu qu'il existait plusieurs grandes familles de ceux-ci. Nous les avons expliquées et une en particulier, les True-SSO. Dans cette section, nous avons distingué les protocoles des implémentations qui sont basées sur ces derniers. Nous avons pu voir que CAS, OpenID, OAuth et SAML étaient des protocoles tandis que Facebook Connect, OneLogin, ECAS et CEF eID étaient des implémentations de ceux-ci.
Dans la seconde section de ce chapitre, nous avons établi différentes comparaisons entre familles de SSO et entre protocoles.

Sur base de ces chapitres, il nous a été demandé de créer un outil permettant de choisir la méthode d'authentification la plus appropriée par rapport à une politique désirée. Cet outil a du être appliqué après à deux cas concrets : l'université de Namur et notre cadre professionnel.
Afin d'avoir un outil simple à utiliser, nous nous sommes orientés vers les arbres de décision. Nous en avons créés un pour le choix des méthodes et un pour le choix des protocoles.
Nous avons pu voir que ces arbres étaient un bon début d'aide à la décision mais qu'ils étaient améliorables. Les différentes remarques les concernant allaient dans le même sens. Les responsable IT à qui nous les avons soumis ont trouvé qu'ils étaient corrects dans l'ensemble mais qu'ils devaient être raffinés. Selon eux, l'arbre des méthodes ne posait pas la bonne question au démarrage et le niveau de sécurité devrait être mis plus en évidence. Quant à l'arbre des protocoles, ils auraient tous deux aimer voir la présence d'un critère supplémentaire concernant les applications hétérogènes ainsi qu'un raffinement des choix avec la présence accrue des Pseudo-SSO.
Nous trouvons qu'ils ont raison et nous avons réalisé une seconde itération en tenant compte de leurs remarques.

Le dernier chapitre que nous avons écrit concernait l'évolution et la conscientisation de la problématique de sécurité autour de l'authentification. Nous avons vu que certains sites collectaient des données personnelles et ne les utilisaient pas toujours à des fins très nobles. Nous avons alors énoncé les directives de l'OCDE et de la Commission Européenne concernant cet aspect de la sécurité.
Nous avons ensuite discuté de l'évolution de l'authentification au niveau des services publiques. Nous avons vu que ceux-ci sont de plus en plus conscients de ce qu'implique la sécurité de leur authentification et des conséquences d'une brèche dans leurs systèmes.
Après cela, une partie a été dédiée aux mobile devices et leur expansion sur le Web. Nous avons conclu qu'ils étaient une bonne opportunité de changer les habitudes des gens et qu'ils pourraient contribuer au remplacement définitif des mots de passe.
Enfin, nous avons terminé par l'évolution de l'authentification et nous avons expliqué les solutions prometteuses qui arrivaient dans le futur. Nous avons décrit l'API en cours de développement par FIDO et W3C, et ce qu'elle pourrait apporter pour la sécurité du Web.

Pour conclure ce document, nous allons parler des difficultés rencontrées lors de la réalisation de ce travail. Nous avons vu que le domaine de l'authentification était très vaste et que le nombre d'articles scientifiques relatant du sujet était conséquent. De ce fait, nous nous sommes confrontés à différents obstacles. Dans un premier temps, nous nous sommes un peu perdus parmi tous ces articles. Il existe plusieurs articles discutant d'un même concept mais différemment. De plus, ils évoquent parfois une version différente de la même méthode d'authentification. Pour prendre un exemple, ce fut le cas avec OpenID 2.0 et OpenID Connect.
Dans un second temps, nous avons eu du mal à prendre du recul et à réaliser une structure cohérence dans ce document. Il n'y avait de fil conducteur et les chapitres ainsi que les sections étaient multiples et mal agencés.

Et en dernier lieu, le travail de relecture fut laborieux. Nous avons fait relire ce mémoire par différentes personnes de notre entourage mais n'étant pas tous informaticien de formation, ils n'ont pas tous pu détecter les erreurs de formulation de texte et de contexte.

Si nous devons recommencer ce mémoire, nous pensons que nous garderions certaines méthodes comme la synthétisation des articles lus. Cependant, nous tenterions de prendre plus de recul et de ne plus foncer tête baissée dans le sujet.

ANNEXE A

Annexe

A.1 Certificat

A.1.1 L'eID : le modèle autrichien

Le système d'authentification par carte d'identité en Autriche se base sur deux parties distinctes, comme décrit par Daniel Slamanig[Slamanig et al., 2014] :

1. L'approche côté client utilisant le carte d'identité.
2. L'approche côté serveur impliquant le smartphone de l'utilisateur.

L'unicité des utilisateurs est réalisée par un nombre unique, appelé sourcePIN. Ce nombre est embarqué dans une structure de données XML, appelée Identity Link ou IDL. Cette structure contient différentes informations telles que le nom, prénom, date de naissance du citoyen autrichien. Elle contient aussi une clé privée permettant d'effectuer des signatures électroniques. L'intégrité et l'authenticité de l'IDL est garantie car elle est signée par la sourcePin Register Authority autrichienne, alias SRA. Le respect de la vie privée est conservé car le gouvernement autrichien a créé une loi , qui oblige d'utiliser le sourcePIN lors de l'identification des utilisateurs auprès des SPs. De ce fait, ils ne recoivent qu'un numéro. Dès lors, la plupart des systèmes eID utilisent des sector-specific PINs (ssPIN), qui sont dérivés du sourcePIN et qui permettent de ne pas lier un nombre à un utilisateur. Cela garantit alors l'intraçabilité entre secteurs. L'accès aux systèmes eID se passent en deux phases, l'enregistrement et l'authentification liée à l'identification.

1. L'enregistrement se fait via email ou dans un bureau communal. Le SRA va alors créer le sourcePIN, l'IDL et générer le certificat. Les données du citoyen vont être ajoutées par la même occasion. La carte sera ensuite envoyée au citoyen.
2. L'authentification est basée sur un software Open Source, le MOA-ID[europaèu, 2015]. Lors de la phase d'authentification, le MOA-ID va alors lire l'IDL de la carte et va le vérifier afin de procéder à l'identification du citoyen. Ensuite, il va demander la création d'une signature digitale qui servera pour l'authentification, qui sera elle aussi vérifiée par les mécanismes classiques (CRL ou OCSP). Le MOA-ID va dériver le sourcePIN pour créer un ssPIN, en regard au SP. Il va ensuite créer une

assertion SAML qui contiendra le ssPIN ainsi que les données utilisateurs et l'envoyer au SP. Le SP le parsera et donnera accès au citoyen à la ressource protégée.

La figure ci-dessous représente le flux des deux étapes précédemment citées.

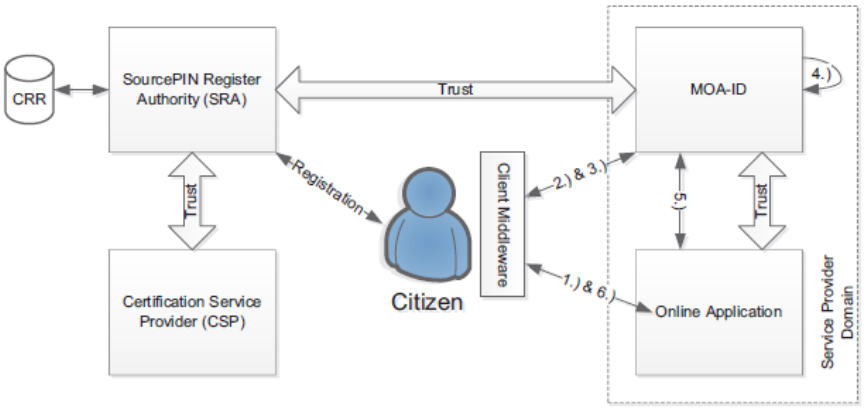

FIGURE A.1 – Système eID autrichien[Slamanig et al., 2014]

A.2 FIDO

A.2.1 U2F

L'image ci-dessous représente l'enregistrement U2F :

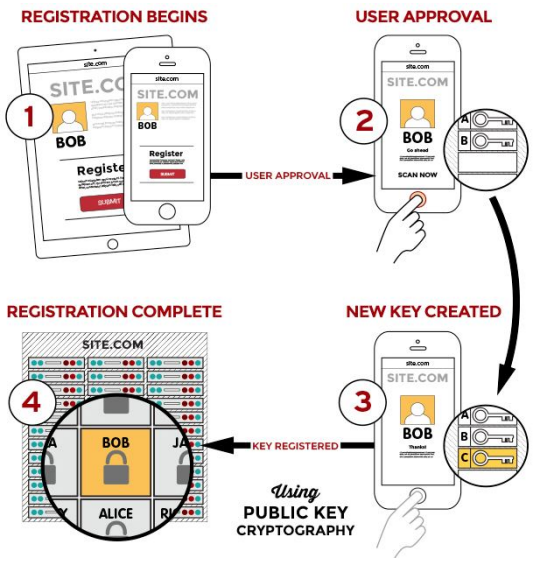

FIGURE A.2 – Enregistrement U2F

L'image ci-dessous représente l'authentification U2F :

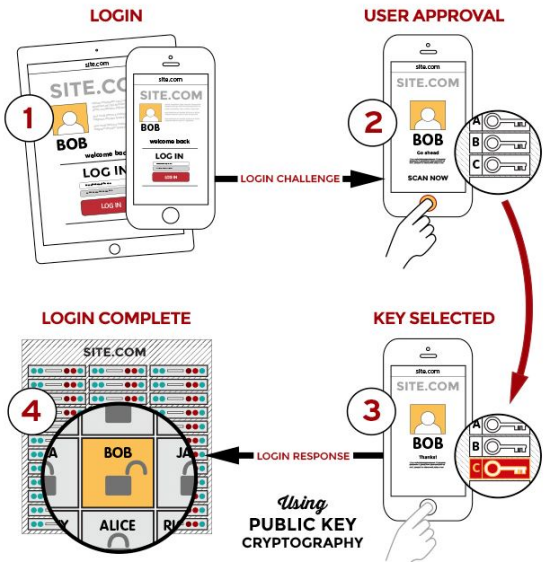

FIGURE A.3 – Authentification renforcée U2F

A.3 OpenID Connect

Les différentes descriptions des flux ci-dessous sont réalisées sur base de la documentation officielle[Nat Sakimura, 2014].

A.3.1 Authorization Code Flow

Property	Authorization Code Flow	Implicit Flow	Hybrid Flow
All tokens returned from Authorization Endpoint	no	yes	no
All tokens returned from Token Endpoint	yes	no	no
Tokens not revealed to User Agent	yes	no	no
Client can be authenticated	yes	no	yes
Refresh Token possible	yes	no	yes
Communication in one round trip	no	yes	no
Most communication server-to-server	yes	no	varies

FIGURE A.4 – Résumé des flow d'authentification OpenID Connect

Voici les huit étapes de ce flow :

1. Le SP prépare une requête d'authentification contenant les paramètres de requêtes désirés.
2. Le SP envoie cette requête à l'IdP.
3. L'IdP obtient le consentement de l'utilisateur.
4. L'IdP redirige l'utilisateur vers le SP avec un code d'autorisation.
5. Le SP demande une du Token EndPoint en utilisant le code reçu.
6. Le SP reçoit la réponse contenant le token ID et le token d'accès dans le corps de la réponse.
7. Le SP valide le token ID et récupère l'identifiant de l'utilisateur.

A.3.2 Implicit Flow

Voici les six étapes de ce flow :

1. Les deux premières étapes sont les mêmes que celles du flow précédent.
2. L'IdP authentifie l'utilisateur.
3. L'IdP obtient le consentement de l'utilisateur.
4. L'IdP redirige l'utilisateur vers le SP avec le token ID ainsi que le token d'accès s'il a été demandé préalablement.
5. Le SP valide le token ID et récupère l'identifiant de l'utilisateur.

A.3.3 Hybrid Flow

Voici les huit étapes de ce flow :

1. Les quatres premières étapes sont les mêmes que l'Implicit Flow.
2. L'IdP redirige l'utilisateur vers le SP avec un token d'autorisation and en fonction du type de réponse, ajoute certain(s) paramètre(s)
3. Le SP demande la réponse du Token Endpoint en utilisant le code reçu.
4. Les trois dernières étapes sont les mêmes que celle de l'Authorization Code Flow.

A.4 SAML

A.4.1 Fédération d'identité

La fédération d'identité permet de transmettre l'identité d'un utilisateur d'un SP à un autre. De la même façon qu'entre IdP et SP, une relation de confiance doit être mise en place entre ces SPs. Il est fort probable que l'utilisateur n'ait pas les mêmes identifiants au sein de ces deux SPs. Chacun d'entre eux pourrait posséder sa propre base de données interne. SAML va permettre de créer une identité fédérée commune entre eux. Lors d'un flux de fédération d'identité, les deux SPs vont se mettre d'accord pour créer un identifiant fédéré. Le principe derrière ce terme est faire un lien entre chacun identifiant local de l'utilisateur présent des deux côtés et un compte qui sera commun aux deux SPs. Cela s'appelle "l'account linking". Une fois l'identifiant créé au niveau du premier SP, il sera partagé au deuxième SP qui le conservera pour les futures connexions.
La figure ci-dessous illustre un exemple concret[OASIS-Open, 2006] :

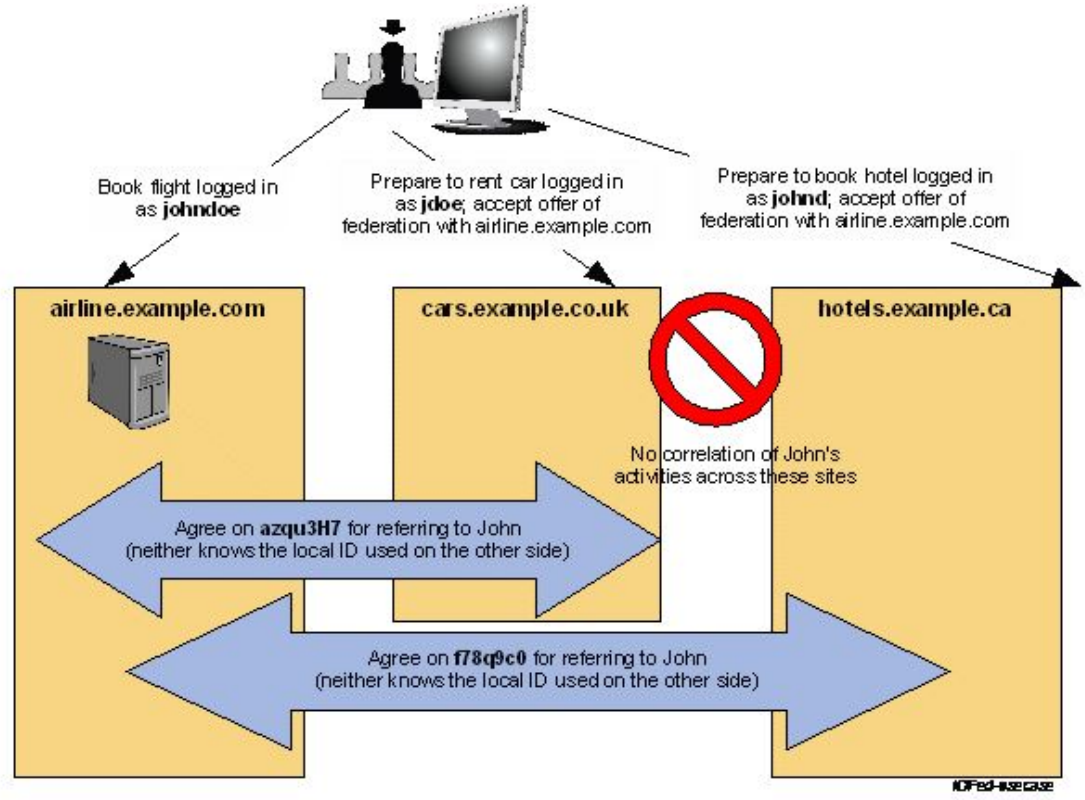

FIGURE A.5 – SAML identité fédérée

Lorsque l'utilisateur accède au site AirlineInc.com, il est connecté sous le compte "johndoe". En cliquant sur un lien , il accèdera au deuxième site CarRentalInc.com. Ce dernier va vérifier les cookies que la navigateur possède et il verra qu'il s'est connecté localement mais qu'il a déjà visité l'IdP (qui est dans ce cas-ci AirlineInc.com). Il demandera le consentement de l'utilisateur afin de créer une identité fédérée localement entre AirlineInc.com et lui-même. S'il accepte, il sera redirigé vers AirlineInc.com où un pseudonyme sera créé . Sur le schéma, il sera appelé "azqu3H7" et sera lié à l'utilisateur local "johndoe". Une fois cette étape réalisée, l'utilisateur sera de nouveau redirigé vers le site CarRentalInc.com avec une assertion qui indiquera que le pseudonyme "azqu3H7" est lié à cet utilisateur et qu'il est bien connecté au niveau de l'IdP. Sachant que CarRentalInc.com ne sait pas à quel utilisateur local ce pseudonyme est lié, il va demander à l'utilisateur de se connecter avec son compte local, le cas échéant, "jdoe". Il pourra ainsi lié le compte local au pseudonyme. Ce changement sera répercuté au niveau de l'IdP pour les futures authentifications. Au final, l'identifiant fédéré a été créé pour le couple d'utilisateurs johndoe et jdoe sur les deux sites.

Le processus expliqué ci-dessus sera appliqué de façon identique pour le troisième site, HotelBooking.com. Il peut d'ailleurs s'appliquer à un grand nombre de sites. De cette façon, il se connectera une fois au niveau de l'IdP et pourra voyager avec une seule identité sur l'ensemble des sites.

A.5 STORK 2.0

Use case des processus métier :

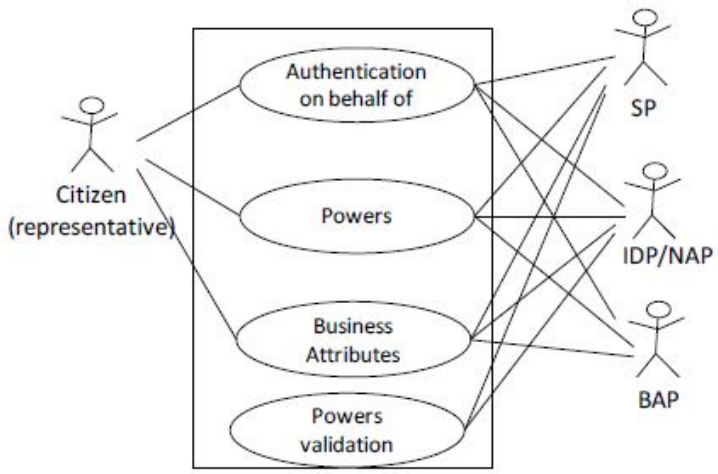

FIGURE A.6 – Use case des processus

A.6 CEF-eID

Evolution du framework STORK vers une solution complète CEF-eID

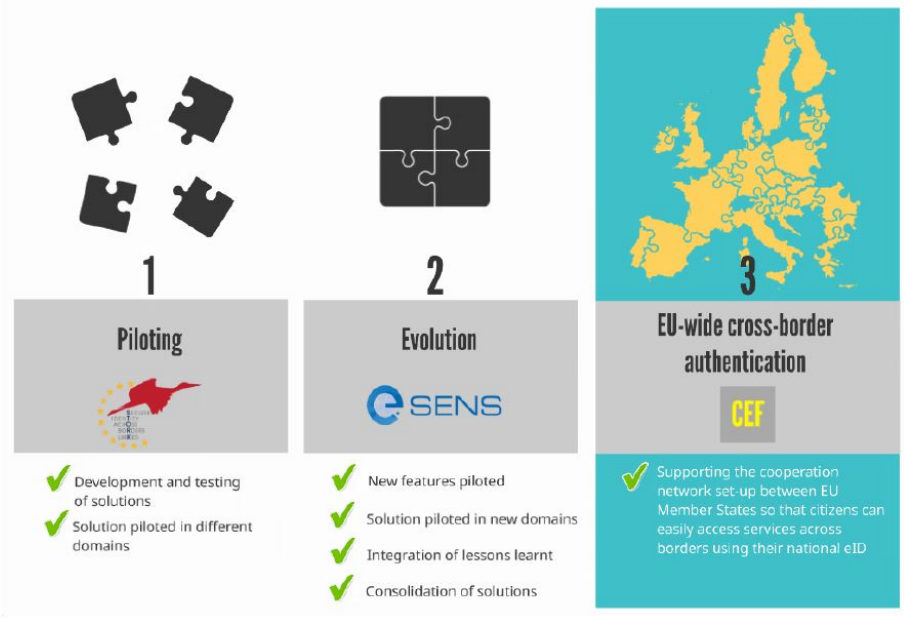

FIGURE A.7 – Evolution de STORK vers CEF eID

A.7 SAML vs OpenID

Tableau de comparaison entre SAML et OpenID Connect[(Novay), 2012] :

Aspect	SAML 2.0	OpenID Connect
"Centricity"	Organization	User (but see below…)
Message format	XML	JSON, REST
Channel	HTTPS	HTTPS
Security	XMLDSig, XMLSEC	HTTPS, JWT
Client registration	Static	Dynamic
IdP discovery	Static	Dynamic
Attribute fetching	Front channel (but see below…)	Back channel

FIGURE A.8 – Comparaison entre SAML et OpenID Connect

Bibliographie

[Alliance, 2014] Alliance, T. F. (2014). Fido 1.0 final specifications have arrived. `https://fidoalliance.org/wp-content/uploads/FIDOMessagingWPv1.pdf`. Accessed on 10.11.2015.

[Andreas Pashalidis, 2004] Andreas Pashalidis, C. J. M. (2004). Impostor a single sign-on system for use from untrusted devices. *Global Telecommunications Conference, 2004. GLOBECOM '04. IEEE*, 4 :2191–2195.

[Apple, 2016a] Apple (2016a). Secure, simple, and even more useful. `http://www.apple.com/apple-pay`. Accessed on 06.04.2016.

[Apple, 2016b] Apple (2016b). Utilisation de la fonctionnalité touch id sur l'iphone et l'ipad. `https://support.apple.com/fr-be/HT201371`. Accessed on 03.05.2016.

[Balliauw, 2009] Balliauw, M. (2009). How we built twittermatic.net. `http://blog.maartenballiauw.be/post/2009/07/02/How-we-built-TwitterMaticnet-Part-4-Authentication-and-membership.aspx`. Accessed on 13.12.2015.

[Belenko and Sklyarov, 2012] Belenko, A. and Sklyarov, D. (2012). Secure password managers and military-grade encryption on smartphones : Oh, really ? `https://media.blackhat.com/bh-eu-12/Belenko/bh-eu-12-Belenko-Password_Encryption-Slides.pdf`. Accessed on 30.10.2015.

[Belgium.be, 2015] Belgium.be (2015). `http://www.taxonweb.be`. Accessed on 29.12.2015.

[Bichsel et al., 2009] Bichsel, P., Camenisch, J., Groß, T., and Shoup, V. (2009). Anonymous credentials on a standard java card. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 600–610, New York, NY, USA. ACM.

[Blizzard, 2015] Blizzard (2015). `http://blizzard.com`. Accessed on 22.11.2015.

[Bonneau et al., 2012] Bonneau, J., Herley, C., Oorschot, P. C. v., and Stajano, F. (2012). The quest to replace passwords : A framework for comparative evaluation of web authentication schemes. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 553–567, Washington, DC, USA. IEEE Computer Society.

[Bonneau et al., 2015] Bonneau, J., Herley, C., van Oorschot, P. C., and Stajano, F. (2015). Passwords and the evolution of imperfect authentication. *Commun. ACM*, 58(7) :78–87.

[Caron, 2011] Caron, S. (2011). Une introduction aux arbres de décision. `https://scaron.info/doc/intro-arbres-decision/intro.pdf`. Accessed on 23.05.2016.

[Colin, 2016] Colin, J.-N. (2015-2016). *Sécurité des systèmes informatiques*, chapter 4, page 4. Faculté d'informatique de l'Unamur.

[CSAM, 2015] CSAM (2015). https://www.csam.be/fr. Accessed on 17.04.2016.

[D. Hardt, 2012] D. Hardt, E. (2012). The oauth 2.0 authorization framework draft-ietf-oauth-v2-31. https://tools.ietf.org/html/draft-ietf-oauth-v2-31#section-1.2. Accessed on 15.12.2015.

[Danny De Cock and Preneel, 2005] Danny De Cock, C. W. and Preneel, B. (2005). The belgian electronic identity card (overview). https://securewww.esat.kuleuven.be/cosic/publications/article-769.pdf. Accessed on 28.09.2015.

[Davenport, 2015] Davenport, S. (2015). Github pushes real buttons to make the internet more secure. http://www.wired.com/2015/10/github-moves-past-password-make-open-source-secure. Accessed on 03.05.2016.

[Egelman, 2013] Egelman, S. (2013). My profile is my password, verify me! : The privacy/convenience tradeoff of facebook connect. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2369–2378, New York, NY, USA. ACM.

[europaèu, 2015] europaèu (2015). Moa-id/sp/ss. https://joinup.ac.europa.eu/asset/moa-idspss/asset_release/moa-id-302. Accessed on 28.09.2015.

[Européenne, 1995] Européenne, C. (1995). Directive 95/46/ce. http://eur-lex.europa.eu/LexUriServ/. Accessed on 04.04.2016.

[Européenne, 2012] Européenne, C. (2012). http://ec.europa.eu/justice/data-protection. Accessed on 04.04.2016.

[Evidian, 2016] Evidian (2016). Web access manager. http://www.evidian.com/fr/. Accessed on 15.01.2016.

[Facebook, 2016] Facebook (2016). Conditions générales. https://www.facebook.com/terms. Accessed on 25.03.2016.

[Fedict, 2014] Fedict (2014). Federal authentication service (fas). http://www.fedict.belgium.be/fr/identification_et_securisation/federal_authentication_service. Accessed on 29.04.2016.

[Florencio and Herley, 2006] Florencio, D. and Herley, C. (2006). Klassp : Entering passwords on a spyware infected machine using a shared-secret proxy. In *Proceedings of the 22Nd Annual Computer Security Applications Conference*, ACSAC '06, pages 67–76, Washington, DC, USA. IEEE Computer Society.

[Florencio and Herley, 2007] Florencio, D. and Herley, C. (2007). A large-scale study of web password habits. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 657–666, New York, NY, USA. ACM.

[Florêncio and Herley, 2008] Florêncio, D. and Herley, C. (2008). One-time password access to any server without changing the server. In *Proceedings of the 11th International Conference on Information Security*, ISC '08, pages 401–420, Berlin, Heidelberg. Springer-Verlag.

[Gibson, 2016] Gibson, S. (2016). Conçu pour la sécurité. https://lastpass.com/fr/enterprise/security/. Accessed on 30.10.2015.

[Google, 2015] Google (2015). https://support.google.com/accounts/answer/1066447?hl=fr. Accessed on 04.10.2015.

[Google, 2016a] Google (2016a). https://apps.google.com/. Accessed on 17.01.2016.

[Google, 2016b] Google (2016b). Google identity platform. https://developers.google.com/identity/. Accessed on 30.01.2016.

[Haller, 1995] Haller, N. M. (1995). The s/key one-time password system.

[Halpin, 2016] Halpin, H. (2016). Web authentication working group charter. https://www.w3.org/2015/12/web-authentication-charter.html. Accessed on 06.04.2016.

[Herbert Leitold, 2015] Herbert Leitold, Antonio Lioy, C. R. (2015). https://www.eid-stork2.eu. Accessed on 18.01.2016.

[Hoffmann, 2015] Hoffmann, M. (2015). Cef eid. https://joinup.ec.europa.eu/software/cefeid/description. Accessed on 19.01.2016.

[HÄRTWICH, 2014] HÄRTWICH, P. (2014). https://www.ffg.at/sites/default/files/downloads/page/140129vienna.pdf. Accessed on 23.12.2015.

[Hubert Le Van Gong, 2015] Hubert Le Van Gong, Dirk Balfanz, A. C. A. B. J. H. (2015). Fido 2.0 : Web api for accessing fido 2.0 credentials. https://www.w3.org/Submission/2015/SUBM-fido-web-api-20151120/. Accessed on 06.04.2016.

[IronKey, 2015] IronKey (2015). http://www.ironkey.com. Accessed on 22.11.2015.

[Jain et al., 2006] Jain, A. K., Ross, A., and Pankanti, S. (2006). Biometrics : A tool for information security. *Trans. Info. For. Sec.*, 1(2) :125–143.

[Jammalamadaka et al., 2006] Jammalamadaka, R. C., Horst, T. W. v. d., Mehrotra, S., Seamons, K. E., and Venkasubramanian, N. (2006). Delegate : A proxy based architecture for secure website access from an untrusted machine. In *Proceedings of the 22Nd Annual Computer Security Applications Conference*, ACSAC '06, pages 57–66, Washington, DC, USA. IEEE Computer Society.

[J.Moore, 2014] J.Moore, R. (2014). Mongodb, tls, and x.509 authentication deep dive. http://www.allanbank.com/blog/security/tls/x.509/2014/10/13/tls-x509-and-mongodb. Accessed on 11.11.2015.

[Kothari, 1985] Kothari, S. C. (1985). Generalized linear threshold scheme. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 231–241, New York, NY, USA. Springer-Verlag New York, Inc.

[Leoutsarakos, 2016] Leoutsarakos, N. (2015-2016). What's wrong with fido? http://zeropasswords.com/pdfs/WHATisWRONG_FIDO.pdf. Accessed on 19.03.2016.

[Lewis and Lewis, 2009] Lewis, K. D. and Lewis, J. E. (2009). Web single sign-on authentication using SAML. *CoRR*, abs/0909.2368.

[Lindemann, 2014] Lindemann, R. (2014). The evolution of authentication. https://fidoalliance.org/wp-content/uploads/2014/12/fido.pdf. Accessed on 10.11.2015.

[Macia, 2013] Macia, P. (2013). http://www.orange-business.com/fr/blogs/securite/webtech/le-saml-le-petit-protocole-qui-monte-qui-monte. Accessed on 22.10.2015.

[Mathieu, 2004] Mathieu, V. (2004). Présentation de cas. https://www.esup-portail.org. Accessed on 04.01.2016.

[Mayer et al., 2014] Mayer, A., Niemietz, M., Mladenov, V., and Schwenk, J. (2014). Guardians of the clouds : When identity providers fail. In *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security*, CCSW '14, pages 105–116, New York, NY, USA. ACM.

[Microsoft, 2016] Microsoft (2016). Oauth 2.0. https://msdn.microsoft.com/en-us/library/hh243647.aspx/. Accessed on 30.01.2016.

[Mohammad Mannan, 2007] Mohammad Mannan, P. C. v. O. (2007). Using a personal device to strengthen password authentication from an untrusted computer. In *Financial Cryptography and Data Security*, 4886, pages 88–103. Springer Berlin Heidelberg.

[Mozilla, 2015] Mozilla (2015). https://mozilla.org. Accessed on 03.10.2015.

[N. Haller, 1998] N. Haller, C. Metz, P. N. M. S. (1998). A one-time password system. https://tools.ietf.org/html/rfc2289. Accessed on 25.09.2015.

[Nat Sakimura, 2014] Nat Sakimura, John Bradley, M. B. J. B. d. M. C. M. (2014). Openid connect core 1.0 incorporating errata set 1. http://openid.net/specs/openid-connect-core-1_0.html#Overview. Accessed on 23.01.2016.

[(Novay), 2012] (Novay), M. O. (2012). Openid connect for surfcone. https://blog.surf.nl/wp-content/uploads/2013/04/SURFnet-OpenID-Connect-1.1-.pdf. Accessed on 27.02.2016.

[OASIS-Open, 2006] OASIS-Open (2006). sstc-saml-tech-overview-2.0-draft-1u. https://www.oasis-open.org/committees/security/docs/. Accessed on 17.01.2016.

[OCDE, 2016] OCDE (2016). Lignes directrices régissant la protection de la vie privée et les flux transfrontières de données de caractère personnel. http://www.oecd.org/fr/internet/ieconomie/lignesdirectricesregissantlaprotectiondelaviepriveeetlesfluxtransfrontieresdedonneesdecaractereper
htm. Accessed on 04.04.2016.

[Parno et al., 2006] Parno, B., Kuo, C., and Perrig, A. (2006). Phoolproof phishing prevention. In *Proceedings of the 10th International Conference on Financial Cryptography and Data Security*, FC'06, pages 1–19, Berlin, Heidelberg. Springer-Verlag.

[Pascal Aubry, 2004] Pascal Aubry, Vincent Mathieu, J. M. (2004). Esup-portail : open source single sign-on with cas (central authentication service). https://www.esup-portail.org/consortium/espace/SSO_1B/cas/eunis2004/cas-eunis2004-article.pdf. Accessed on 12.01.2016.

[Pashalidis and Mitchell, 2003] Pashalidis, A. and Mitchell, C. J. (2003). A taxonomy of single sign-on systems. In *Proceedings of the 8th Australasian Conference on Information Security and Privacy*, ACISP'03, pages 249–264, Berlin, Heidelberg. Springer-Verlag.

[Reinke, 2013] Reinke, J. (2013). Comprendre oauth2. http://www.bubblecode.net/fr/2013/03/10/comprendre-oauth2/. Accessed on 04.11.2015.

[Robinson and Bonneau, 2014] Robinson, N. and Bonneau, J. (2014). Cognitive disconnect : Understanding facebook connect login permissions. In *Proceedings of the Second ACM Conference on Online Social Networks*, COSN '14, pages 247–258, New York, NY, USA. ACM.

[RSA, 2015] RSA (2015). http://www.emc.com/security/rsa-securid/index.htm. Accessed on 22.11.2015.

[Salah Machani, 2014] Salah Machani, Rob Philpott, S. S. J. K. J. H. (2014). Fido uaf architectural overview. https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-uaf-overview-v1.0-ps-20141208.pdf. Accessed on 10.11.2015.

[Sampath Srinivas, 2015] Sampath Srinivas, Dirk Balfanz, E. T. A. C. (2015). Universal 2nd factor (u2f) overview. https://fidoalliance.org/specs/fido-u2f-overview-ps-20150514.pdf. Accessed on 10.11.2015.

[SANS-Institute, 2002] SANS-Institute (2002). Biometrics and user authentication. https://www.sans.org/reading-room/whitepapers/authentication/biometrics-user-authentication-122. Accessed on 03.03.2016.

[shyong Tsai et al., 2006] shyong Tsai, C., chi Lee, C., and shiang Hwang, M. (2006). Password authentication schemes : Current status and key issues. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.100.8931. Accessed on 25.10.2015.

[Silver et al., 2014] Silver, D., Jana, S., Boneh, D., Chen, E., and Jackson, C. (2014). Password managers : Attacks and defenses. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 449–464, San Diego, CA. USENIX Association.

[Slamanig et al., 2014] Slamanig, D., Stranacher, K., and Zwattendorfer, B. (2014). User-centric identity as a service-architecture for eids with selective attribute disclosure. In *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, SACMAT '14, pages 153–164, New York, NY, USA. ACM.

[Sun and Beznosov, 2012] Sun, S.-T. and Beznosov, K. (2012). The devil is in the (implementation) details : An empirical analysis of oauth sso systems. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 378–390, New York, NY, USA. ACM.

[Sun et al., 2010] Sun, S.-T., Boshmaf, Y., Hawkey, K., and Beznosov, K. (2010). A billion keys, but few locks : The crisis of web single sign-on. In *Proceedings of the 2010 Workshop on New Security Paradigms*, NSPW '10, pages 61–72, New York, NY, USA. ACM.

[Sun et al., 2011] Sun, S.-T., Pospisil, E., Muslukhov, I., Dindar, N., Hawkey, K., and Beznosov, K. (2011). What makes users refuse web single sign-on? : An empirical investigation of openid. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, SOUPS '11, pages 4 :1–4 :20, New York, NY, USA. ACM.

[Wang et al., 2011] Wang, N., Xu, H., and Grossklags, J. (2011). Third-party apps on facebook : Privacy and the illusion of control. In *Proceedings of the 5th ACM Symposium on Computer Human Interaction for Management of Information Technology*, CHIMIT '11, pages 4 :1–4 :10, New York, NY, USA. ACM.

[Wikipedia, 2015] Wikipedia (2015). https://fr.wikipedia.org/wiki/GrIDsure. Accessed on 29.09.2015.

[Wikipedia, 2016a] Wikipedia (2016a). Authentification forte. https://fr.wikipedia.org/wiki/Authentification_forte. Accessed on 02.03.2016.

[Wikipedia, 2016b] Wikipedia (2016b). Directive 95/46/ce sur la protection des données personnelles. https://fr.wikipedia.org/wiki/Directive-95/46/CE-sur-la-protection-des-donnees-personnelles. Accessed on 04.04.2016.

[Yahoo, 2016a] Yahoo (2010-2016a). http://www.onelogin.com/. Accessed on 15.01.2016.

[Yahoo, 2016b] Yahoo (2016b). Authentication and authorization with yahoo! https://developer.yahoo.com/auth/. Accessed on 30.01.2016.

[Yubico, 2016] Yubico (2016). YubiKey 4 & YubiKey 4 Nano. https://www.yubico.com/products/yubikey-hardware/yubikey4. Accessed on 12.02.2016.